TUGAS AKHIR – KI141502

IMPLEMENTASI ALGORITMA GREEDY UNTUK OPTIMASI AKSI COMPUTER PLAYER DALAM PERMAINAN KARTU CAPSA BANTING

EVAN BANGUN
NRP 05111440000169

Dosen Pembimbing
Imam Kuswardayan, S.Kom, M.T.
Dr.Eng. Nanik Suciati, S.Kom, M.Kom.

DEPARTEMEN INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN KOMUNIKASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2018

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR – KI141502

IMPLEMENTASI ALGORITMA GREEDY UNTUK OPTIMASI AKSI COMPUTER PLAYER DALAM PERMAINAN KARTU CAPSA BANTING

EVAN BANGUN
NRP 05111440000169

Dosen Pembimbing
Imam Kuswardayan, S.Kom, M.T.
Dr.Eng. Nanik Suciati, S.Kom, M.Kom

DEPARTEMEN INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN KOMUNIKASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2018

[Halaman ini sengaja dikosongkan]

FINAL PROJECT – KI141502

IMPLEMENTATION OF GREEDY ALGORITHM TO OPTIMIZE COMPUTER PLAYER ACTION IN BIG 2 CARD GAME

EVAN BANGUN
NRP 05111440000169

Advisor
Imam Kuswardayan, S.Kom, M.T.
Dr.Eng. Nanik Suciati, S.Kom, M.Kom.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2018

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

IMPLEMENTASI ALGORITMA GREEDY UNTUK OPTIMASI AKSI COMPUTER PLAYER DALAM PERMAINAN KARTU CAPSA BANTING

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada Bidang Studi Interaksi Grafika dan Seni
Program Studi S-1 DepartemenInformatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh :
EVAN BANGUN
NRP : 05111440000169

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Imam Kuswardayan, S.Kom, M.T. .....
NIP: 197612152003121001 (pembimbing 1)

Dr.Eng. Nanik Suciati, S.Kom, M.Kom. .....
NIP: 197104281994122001 (pembimbing 2)

SURABAYA
APRIL 2018

[Halaman ini sengaja dikosongkan]

IMPLEMENTASI ALGORITMA GREEDY UNTUK OPTIMASI AKSI COMPUTER PLAYER DALAM PERMAINAN KARTU CAPSA BANTING

Nama Mahasiswa	: EVAN BANGUN
NRP	: 05111440000169
Departemen	: Informatika FTIK-ITS
Dosen Pembimbing I	: Imam Kuswardayan, S.Kom, M.T.
Dosen Pembimbing II	: Dr.Eng. Nanik Suciati, S.Kom, M.Kom.

ABSTRAK

Permainan kartu adalah jenis permainan yang sudah lama ada dan tidak asing lagi bagi seluruh kalangan masyarakat, baik *trading cards* ataupun *playing cards*. Permainan kartu ini pada umumnya dimainkan dengan lebih dari 1 orang. Tetapi seiring berkembangnya teknologi permainan kartu ini menjadi mungkin dimainkan seorang diri dengan munculnya program yang dapat memainkan suatu permainan dengan aksi tertentu atau lebih dikenal dengan *Computer Player*. Banyak algoritma yang bisa diterapkan pada *Computer Player* ini tergantung dari permainan yang akan dimainkan. Contoh permainan kartu yang bisa menjadi bisa dimainkan seorang diri karena adanya *Computer Player* adalah permainan kartu Capsa Banting.

Pada penelitian ini penulis menawarkan sebuah cara penggunaan algoritma pemilihan kartu yang dapat digunakan untuk menentukan kartu terbaik yang bisa dikeluarkan oleh *Computer Player*. "Capsa banting" merupakan permainan kartu yang dirancang untuk smartphone berbasis android. Permainan ini menerapkan peraturan dasar permainan kartu pada umumnya, yaitu menghabiskan kartu di tangan secepat mungkin untuk memenangkan permainan. Oleh karena itu dibutuhkan sebuah cara agar kartu yang dikeluarkan adalah kartu terbaik yang dapat dikeluarkan. Untuk pemilihan kartu tersebut maka digunakan algoritma Greedy.

Untuk uji coba digunakan 4 pemain *Computer Player*, yang masing-masing menggunakan algoritma Greedy, yang akan memainkan permainan sebanyak 30 kali dimana setiap *Computer Player* memiliki tingkat kesulitan yang berbeda-beda. Terdapat 3 tingkat kesulitan yang diterapkan untuk Computer Playernya yaitu *easy*, *medium*, dan *hard*. Hasilnya, *Computer Player* selalu mengeluarkan kartu yang terbaik yang dapat dikeluarkan pada tiap putarannya dengan hasil akhir *Computer Player* tingkat kesulitan *hard* memenangkan permainan lebih banyak daripada *easy* dan *medium*. Oleh karena itu dapat dikatakan bahwa algoritma Greedy ini bisa diterapkan dalam permainan kartu Capsa Banting.

Kata kunci: Greedy, capsa banting, Computer Player

IMPLEMENTATION OF GREEDY ALGORITHM TO OPTIMIZE COMPUTER PLAYER ACTION IN BIG 2 CARD GAME

Name	: EVAN BANGUN
NRP	: 05111440000169
Major	: Informatics– FTIK ITS
Supervisor I	: Imam Kuswardayan, S.Kom, M.T.
Supervisor II	: Dr.Eng. Nanik Suciati, S.Kom, M.Kom.

ABSTRACT

*Card games is a game that has long existed and familiar to all circles of society, whether it is trading cards or playing cards.Cards games are usually played by more than 1 person or player. But as technology advances theses card games become possible to played alone with the appearance of a program which can play a game with particular action, better known as Computer Player. There are a lot of algorithms that can be implemented in these Computer Players depending on the games to beplayed. An example of a card game which can be played alone with the help of Computer Players is Big 2 card game. "*Capsa Banting"* is a card game designed for an Android-based Smartphone. This game implements the basic rules of card games in general, which is to spend the card in hand as fast as possible and win the game.*

In this study the author offers a way of using a card-selecting algorithm which can be used to determine the best card the Computer Player can play. Therefore it is crucial to make sure that the card played is the best card possible. Thus for the card selection, Greedy Algorithm is used

Trials will use 4 Computer Players with Greedy algorithm implemented, which has different difficulty level, playing 30 matches. There are 3 levels of difficulties implemented in the game, which are easy, medium, and hard. The results are, the Computer Players successfully played the best card possible

on each turn of the game, which ultimately leading to the Computer Player with hard difficulty implemented, winning more matches than the Computer Players with easy or medium difficulties.

Keywords: Greedy, capsa banting, Computer Player

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yesus Kristus atas pimpinan, penyertaan, dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul :

IMPLEMENTASI ALGORITMA GREEDY UNTUK OPTIMASI AKSI COMPUTER PLAYER DALAM PERMAINAN KARTU CAPSA BANTING

Pengerjaan Tugas Akhir ini dilakukan untuk memenuhi salah satu syarat meraih gelar Sarjana di Departemen Informatika Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember.

Dengan selesainya Tugas Akhir ini, diharapkan apa yang telah dikerjakan penulis dapat memberikan manfaat bagi perkembangan ilmu pengetahuan, terutama di bidang teknologi informasi serta bagi diri penulis sendiri selaku pengerja.

Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Tuhan Yesus Kristus YME.
2. Orang tua, Abang,serta Saudara-saudara yang selalu mendoakan dan mendukung penulis.
3. Pak Imam Kuswardayan, S.Kom, M.T.selaku pembimbing I yang selalu memberikan arahan, motivasi dan bantuan sekaligus bimbingan kepada penulis selama pengerjaan Tugas Akhir.
4. IbuDr.Eng. Nanik Suciati, S.Kom, M.Kom.selaku pembimbing II yang juga telah sangat membantu, dan membimbing saat pengerjaan Tugas Akhir ini.
5. Bapakdan Ibu Dosen Karyawan Teknik Informatika FTIK-ITS yang telah memberikan ilmunya.
6. Teman-teman angkatan 2014 yang telah membantu, berbagi ilmu, menjaga kebersamaan, dan memberi motivasi kepada penulis, kakak-kakak angkatan 2013, 2012, serta adik-adik

angkatan 2015 dan 2016 yang membuat penulis untuk selalu belajar.

7. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depannya.

Surabaya, April 2018

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Bab ini membahas garis besar penyusunan tugas akhir yang meliputi latar belakang, tujuan pembuatan, rumusan dan batasan permasalahan, metodologi penyusunan tugas akhir, dan sistematika penulisan.

1.1. Latar Belakang

Salah satu cara untuk beristirahat atau relaksasi adalah dengan cara bermain game (permainan). Dari banyak jenis permainan yang ada, kartu adalah satu permainan terpopuler yang bisa dimainkan di waktu luang. Permainan kartu juga bergantung pada jenis kartu yang dimainkan, ada permainan kartu dengan menggunakan trading card (Magic: The Gathering, Yu-Gi-Oh!, Vanguard) dan ada yang menggunakan kartu remi. Salah satu dari banyak permainan yang bisa di mainkan dengan kartu remi ini adalah permainan capsa banting.

Capsa banting hanya bisa dimainkan oleh 4 orang, tidak bisa kurang dan tidak bisa lebih. Dalam permainan capsa banting tidak digunakan kartu joker jadi kartu yang di gunakan hanya berjumlah 52 yang akan dibagikan sampai habis ke 4 pemain tersebut, dimana setiap pemain akan memulai permainan dengan 13 kartu di tangan masing-masing. Pemain dapat mengeluarkan kartu di tangan mereka satu persatu atau bisa pula mengkombinasikannya. Ada beberapa kombinasi kartu antara lain, kombinasi 2 kartu sama (pair), kombinasi 3 kartu yang sama (threes), serta kombinasi 5 kartu yang jika diurutkan dari terkecil antara lain, straight, flush, full house, four of a kind, straight flush, dan royal straight flush. Pemain yang pertama kali menghabiskan seluruh kartu ditangannya adalah pemenangnya. Agar dapat menghabiskan kartu di tangan dengan cepat dan menang maka pemain harus bisa memilih kartu mana yang paling baik mereka keluarkan untuk setiap putarannya.

Untuk mengatasi masalah tersebut, maka dapat digunakan metode optimasi untuk menentukan pilihan kartu terbaik yang bisa digunakan untuk menimpa kartu yang dikeluarkan pemain sebelumnya. Optimasi itu sendiri adalah suatu proses untuk mencapai hasil yang ideal atau optimasi (nilai efektif yang dapat dicapai). Optimasi dapat diartikan sebagai suatu bentuk mengoptimalkan sesuatu hal yang sudah ada, ataupun merancang dan membuat sesusatu secara optimal. Jika di terapkan pada permainan capsa banting maka optimasi bisa diartikan sebagai metode pemilihan kartu terbaik yang dapat dimainkan setiap putaran agar dapat menghasilkan kondisi terbaik, dalam hal ini, memenangkan permainan.

Dalam tugas akhir ini, akan digunakan metode optimasi Greedy. Greedy itu sendiri adalah sebuah algoritma penyelesaian masalah dimana dilakukan pencarian pilihan terbaik secara lokal pada setiap langkah-langkahnya dengan harapan menemukan pilihan terbaik secara global[1]. Sehingga dalam kasus ini kartu yang akan dipilih adalah kartu terendah yang bisa dikeluarkan untuk mengalahkan kartu yang sudah di keluarkan pemain sebelumnya. Sehingga jika awalnya kartu yang dikeluarkan adalah random yang dapat menyisakan kartu kartu rendah sehingga menyulitkan pemain untuk menang, dengan menggunakan algoritma greedy sudah dapat dipastikan bahwa kartu yang tersisa adalah kartu bernilai tinggi karena semua kartu rendah yang mungkin dikeluarkan telah dikeluarkan. Dengan metode ini maka tingkat kesulitan *Artifical Intelligence* (atau biasa disebut AI) pun akan meningkat dan membuat permainan lebih menantang.

1.2. Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

a. Bagaimana merancang skenario dan tingkat kesulitan pada permainan ?
b. Bagaimana cara mengimplementasi algoritma Greedy pada *Computer Player*?

1.3. Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan antara lain:

a. Permainan harus dimainkan oleh 4 pemain.
b. Permainan berbasis mobile (android / iOS)
c. Lingkungan pengembangan yang digunakan menggunakan aplikasi Unity 5.6.3p2 (64-bit) Free License dan bahasa pemrograman C#.

1.4. Tujuan

Tujuan dari pembuatan tugas akhir ini antara lain :

a. Membuat aplikasi permainan berbasis mobile yang memiliki *Computer Player* dengan kesulitan tersendiri.
b. Pemanfaatan metode Greedy dalam pembuatan *Computer Player* pada aplikasi permainan.

1.5. Manfaat

Manfaat dari tugas akhir ini adalah terciptanya aplikasi permainan kartu capsa banting dengan *Computer Player* yang memliki tingkat kesulitan tersendiri yang bisa membuat pemain tidak bosan. Hasil penelitian dengan metode Greedy ini diharapkan kedepannya dapat diterapkan sebagai dasar dari pengembangan permainan-permainan kartu lain yang serupa atau lebih kompleks.

1.6. Metodologi Pembuatan Tugas Akhir

Pembuatan tugas akhir dilakukan menggunakan metodologi sebagai berikut:

A. Studi literatur
   Dalam studi literatur penulis akan mempelajari beberapa referensi terkait topik tugas akhir. Beberapa referensi tersebut adalah mengenai Unity, pemrograman C#, serta algoritma optimasi Greedy.

B. Analisis dan desain perangkat lunak

Aplikasi ini memiliki fitur-fitur sebagai berikut :

1. Bermain sesuai dengan aturan capsa banting pada umumnya dan sesuai dengan yang sudah di jelaskan
2. Permainan terdiri atas beberapa tingkat kesulitan sesuai yang dijabarkan diatas

Desain perangkat lunak mencakup pengumpulan data yang dibutuhkan pada metode yang diterapkan.

C. Perancangan dan implementasi perangkat lunak
Aplikasi ini akan dibangun menggunakan Unity Game Engine versi 5.6.3p2 (64-bit), dengan bahasa pemrograman C#, serta algoritma optimasi greedy yang di terapkan pada *Computer Player*.

D. Pengujian dan Evaluasi
Tahap pengujian dan evaluasi berisi pengujian aplikasi dan evaluasi berdasarkan hasil pengujian. Pada tahap ini dilakukan pengujian dari fitur perangkat lunak, apakah sesuai dengan yang diharapkan serta tidak terdapat bug. Selain itu dilakukan juga pengujian terhadap algoritma. Apakah pilihan kartu yang dikeluarkan sudah sesuai dengan algoritma Greedy yang diterapkan.

E. Penyusunan laporan tugas akhir
Pada tahap ini dilakukan penyusunan laporan yang berisi dasar teori, dokumentasi dari perangkat lunak, dan hasil-hasil yang diperoleh selama pengerjaan tugas akhir.

1.7. Sistematika Penulisan

Buku Tugas Akhir ini terdiri atas beberapa bab yang tersusun secara sistematis, yaitu sebagai berikut.

1. BAB 1, Pendahuluan, menjelaskan latar belakang, batasan masalah, tujuan dari pembuatan tugas akhir ini serta metodologi yang digunakan selama penyusunan.
2. BAB 2, Tinjauan Pustaka, memaparkan hasil studi literatur yang digunakan sebagai dasar untuk menyelesaikan tugas akhir ini, terdiri atasdeskripsi mengenai perancangan perangkat lunak, *human computer interaction*, aturan

permainan kartu capsa banting, UNITY sebagai game engine yang digunakan, bahasa pemrograman C#, dan algoritma Greedy sebagai algoritma yang digunakan.

3. BAB 3, Analisa dan Perancangan sistem game yang dikembangkan. Pada tahap ini dijelaskan deskripsi dari game Capsa Banting dan dianalisa bagaimana *gameplay* dari game Capsa Banting. Setelahnya dibahas menganai bagaimana perancangan *gameplay*, perancangan tingkat kesulitan, dan bagaiman algoritma Greedy diterapakan dan disesuaikan dengan aturan permainan Capsa Banting.
4. BAB 4, Bab ini membahas implementasi dari rancangan sistem yang dilakukan pada tahap perancangan. Penjelasan implementasi meliputi implementasi pembuatan aplikasi permain dengan menerapkan algoritma Greedy yang disesuaikan dengan aturan permainan.
5. BAB 5, Pengujian dan Evaluasi, pengujian dilakukan dengan sampel 30 permainan yang dimainkan oleh 4 *Computer Player* yang masing-masing menerapkan algoritma Greedy tetapi dengan tingkat kesulitan yang berbeda. Akan diamati apakah *Computer Player* benar-benar memilih kartu terbaik menurut pemilih Greedy dan apakah tingkat kesulitan yang lebih tinggi dapat memenangkan lebih banyak permainan.
6. BAB 6, Kesimpulan dan Saran, berisi tentang kesimpulan yang didapat dari proses pembuatan tugas akhir beserta saran-saran untuk pengembangan selanjutnya.

[Halaman ini sengaja dikosongkan]

BAB II
TINJAUAN PUSTAKA

Bab ini membahas teori-teori yang mendukung pembuatan tugas akhir. Teori yang mendukung tersebut adalah deskripsi mengenai perancangan perangkat lunak, *human computer interaction*, Unity sebagai *game engine*, bahasa pemrograman C#, aturan permainan capsa banting, dan algoritma Greedy.

2.1. Rancang Bangun Perangkat Lunak

Pengertian rancang bangun perangkat lunak adalah suatu kegiatan menerjemahkan hasil analisa ke dalam bentuk paket perangkat lunak kemudian menciptakan sistem tersebut ataupun memperbaiki sistem yang sudah ada.

2.2. Human Computer Interaction

Interaksi manusia dan komputer (bahasa Inggris: *human–computer interaction*, HCI) adalah disiplin ilmu yang mempelajari hubungan antara manusia dan komputer yang meliputi perancangan, evaluasi, dan implementasi antarmuka pengguna komputer agar mudah digunakan oleh manusia. Ilmu ini berusaha menemukan cara yang paling efisien untuk merancang pesan elektronik. Sedangkan interaksi manusia dan komputer sendiri adalah serangkaian proses, dialog dan kegiatan yang dilakukan oleh manusia untuk berinteraksi dengan komputer yang keduanya saling memberikan masukan dan umpan balik melalui sebuah antarmuka untuk memperoleh hasil akhir yang diharapkan.[2].

2.3. Unity (Game Engine)

Unity adalah sebuah game engine yang berfungsi untuk menghasilkan sebuah game 3D/2D, dalam berbagai genre, dan untuk berbagai platform dengan sangat mudah dan cepat. Sampai saat ini Unity telah mendukung lebih dari 5 platform utama mulai dari desktop, Web, Android, IOS, XBOX, Nintendo Wii sampai

PS4. Keunggulan Unity terletak dari mekanisme development game yang jauh lebih ringkas jika dibandingkan dengan menggunakan game engine lainnya.[3].

2.4. Bahasa Pemrograman C#

Bahasa C# adalah sebuah bahasa pemrograman modern yang bersifat general-purpose, berorientasi objek, yang dapat digunakan untuk membuat program di atas arsitektur Microsoft .NET Framework. Bahasa C# ini memiliki kemiripan dengan bahasa Java, C dan C++ (selengkapnya dapat dilihat pada Sejarah Bahasa C#).

Bahasa pemrograman ini dikembangkan oleh sebuah tim pengembang di Microsoft yang dipimpin oleh Anders Hejlsberg, seorang yang telah lama malang melintang di dunia pengembangan bahasa pemrograman karena memang ialah yang membuat Borland Turbo Pascal, Borland Delphi, dan juga Microsoft J++.

Kini, C# telah distandarisasi oleh European Computer Manufacturer Association (ECMA) dan juga International Organization for Standardization (ISO) dan telah menginjak versi 3.0 yang mendukung beberapa fitur baru semacam Language Integrated Query (LINQ) dan lain-lainnya.[4]

2.5. Aturan Permainan Capsa Banting

Secara umum, aturan bermain capsa banting adalah sebagai berikut:

1) Urutan lambang menurut yang terbesar hingga yang terkecil adalah : Spade > Heart > Club > Diamond. Sedangkan untuk nilai kartu nya 2 > As > King > Queen > Jack > 10 > 9 > 8 > 7 > 6 > 5 > 4 > 3.
2) Permainan dimulai dengan setiap pemain memiliki 13 kartu di tangan. (52 kartu dibagi ke keempat pemain secara rata sampai habis)
3) Pemain yang memiliki kartu 3 Diamond harus mengeluarkan kartu pertama kali (memulai permainan). Kartu yang dikeluarkan boleh berupa single (1 kartu) atau

dikombinasikan sebisanya selama angka 3 Diamond termasuk.

4) Kartu kombinasi (paket) hanya bisa ditimpa oleh paket dengan jumlah kartu sama (pair hanya dilawan dengan pair, threes hanya bisa dilawan dengan threes, dan 5 kartu hanya bisa dilawan dengan 5 kartu).
5) Urutan Kombinasi 5 kartu dari yang terbesar adalah : Royal Straight Flush (Urutan kartu As-King-Queen-Jack-10 bersimbol spade) > Straight Flush (Urutan 5 kartu bersimbol sama) > Four of a Kind (4 Kartu angka sama + 1 kartu bebas) > Full House (Gabungan Threes + Pair) > Flush (5 kartu bebas bersimbol sama) > Straight (5 kartu berurutan).
6) Pemain yang mengeluarkan kartu tertinggi dalam suatu putaran memiliki hak untuk memulai putaran berikutnya dan bisa mengeluarkan kartu apapun secara bebas.

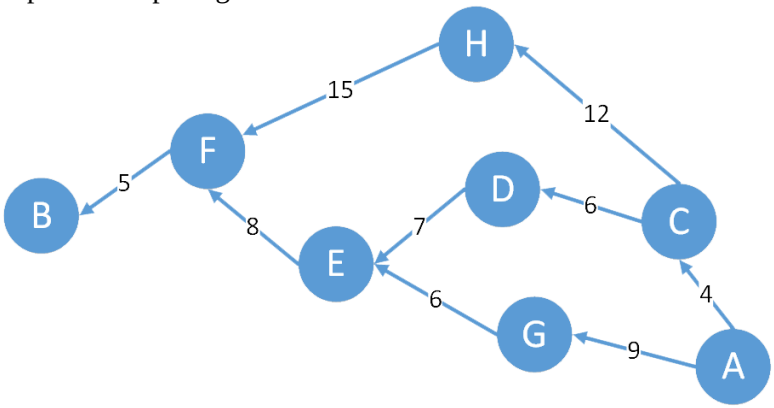
Pemain yang paling pertama menghabiskan kartu ditangannya adalah pemenangnya. Strategi pada permainan ini adalah pemain harus bisa memilih kapan harus mengeluarkan kartu yang lebih tinggi daripada pemain sebelumnya atau kapan harus melewatkan giliran untuk menyusun strategi di giliran berikutnya. Sehingga saat dibutuhkan pemain dapat mengeluarkan kartu yang diperlukan dan memenangkan permainan.

2.6. Algoritma Greedy

Algoritma greedy merupakan jenis algoritma yang menggunakan pendekatan penyelesaian masalah dengan mencari nilai maksimum sementara pada setiap langkahnya. Nilai maksimum sementara ini dikenal dengan istilah local maximum. Pada kebanyakan kasus, algoritma greedy tidak akan menghasilkan solusi paling optimal, begitupun algoritma greedy biasanya memberikan solusi yang mendekati nilai optimum dalam waktu yang cukup cepat.

Contoh penyelesaian masalah dengan algoritma greedy antara lain mencari jarak terpendek dari peta. Misalkan ingin

bergerak dari titik A ke titik B, dan terdapat beberapa jalur seperti dapat dilihat pada gambar 2.1.



Gambar 2.1 Contoh kasus algoritma Greedy

Untuk mencari jarak terpendek dari A ke B, sebuah algoritma greedy akan menjalankan langkah-langkah seperti berikut:

1. Kunjungi satu titik pada graph, dan ambil seluruh titik yang dapat dikunjungi dari titik sekarang.
2. Cari *local maximum* ke titik selanjutnya.
3. Tandai graph sekarang sebagai graph yang telah dikunjungi, dan pindah ke *local maximum* yang telah ditentukan.
4. Kembali ke langkah 1 sampai titik tujuan didapatkan.

Jika mengaplikasikan langkah-langkah di atas pada graph A ke B sebelumnya maka kita akan mendapatkan pergerakan seperti berikut:

1. Mulai dari titik awal (A). Ambil seluruh titik yang dapat dikunjungi.
2. Local maximum adalah ke C, karena jarak ke C adalah yang paling dekat.
3. Tandai A sebagai titik yang telah dikunjungi, dan pindah ke C.
4. Ambil seluruh titik yang dapat dikunjungi dari C.
5. Local maximum adalah ke D, dengan jarak 6.

6. Tandai C sebagai titik yang telah dikunjungi, dan pindah ke D.

Masing-masing titik dari D sampai dengan F hanya memiliki 1 titik yang dapat di kunjungi, oleh karena itu dengan menggunakan algoritma greedy pada graph di atas, hasil akhir yang akan didapatkan sebagai jarak terpendek adalah A-C-D-E-F-B. Hasil jarak terpendek yang didapatkan ini tidak tepat dengan jarak terpendek yang sebenarnya (A-G-E-F-B). Algoritma greedy memang tidak selamanya memberikan solusi yang optimal, dikarenakan pencarian *local maximum* pada setiap langkahnya, tanpa memperhatikan solusi secara keseluruhan.[5]

[Halaman ini sengaja dikosongkan]

BAB III
ANALISIS DAN PERANCANGAN SISTEM

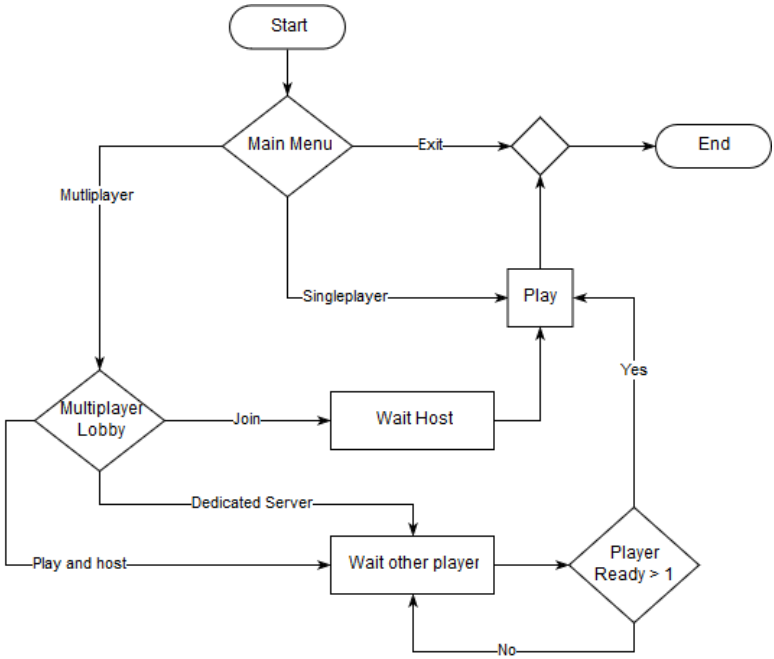
Bab ini membahas tahap analisis permasalahan dan perancangan tugas akhir. Pada bagian awal akan dibahas mengenai *gameplay* atau aturan dari game 'Capsa Banting'. Selajutnya dibahas mengenai penggunaan algoritma Greedy yang digunakan dalam pemilihan kartu yang akan dimainkan *Computer Player*. Algoritma Greedy yang digunakan pada game Capsa Banting telah disesuaikan dengan aturan permainan capsa banting agar dapat mengeluarkan kartu terbaik dan sesuai dengan aturan.

3.1. Analisis

3.1.1. Deskripsi Game Capsa Banting

Capsa Banting adalah jenis permainan kartu yang menggunakan kartu bermain (playing cards) dengan peraturan yang sama persis dengan permainan kartu capsa banting pada umumnya. Dalam Bahasa inggris permainan ini lebih dikenal dengan nama *Big 2*. Inti dari permainan ini adalah pemain harus menghabiskan kartu di tangan secepat mungkin untuk dapat memenangkan permainan. Perbedaan permainan kartu ini dengan beberapa permainan kartu lain yang sejenisnya adalah pada permainan ini pemain dapat mengeluarkan lebih dari 1 kartu secara bersamaan atau lebih sering disebut paket. Pemain harus bisa memilih waktu yang tepat untuk mengeluarkan paket-paket tersebut agar bisa menghabiskan kartu di tangan dengan cepat. Dalam permainan kartu capsa banting ini terdapat 2 mode permainan yaitu s*ingleplayer* dan m*ultiplayer*, dimana pada mode *singleplayer* pemain bermain melawan 4 *Computer Player* lain sementara pada *multiplayer* pemain dapat bermain dengan pemain lain. Tetapi karena permainan capsa banting ini hanya bisa dimainkan oleh 4 orang, maka jika jumlah pemain tidak mencukupi maka akan bermain melawan *Computer Player* juga.

Untuk alur permainan secara garis besar dapat dilihat pada gambar 3.1

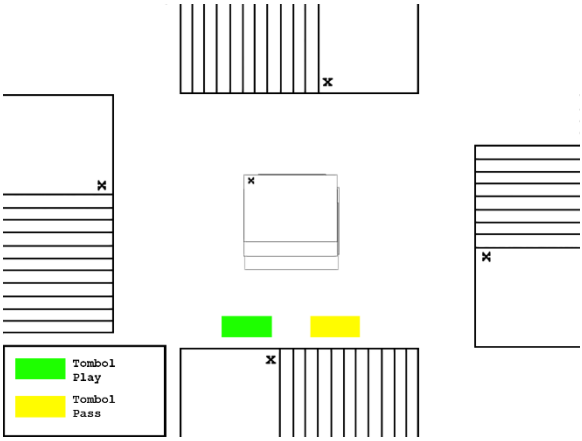


Gambar 3.1 Garis Besar Alur Pemainan Capsa Banting

3.1.2. Analisis *Gameplay* Aturan Permainan

Capsa Banting memiliki aturan yang cukup sederhana. Setiap pemain memulai permainan dengan 13 kartu ditangan. Setelah itu pemain yang memiliki kartu 3 diamond di tangan akan membuka permainan. Pemain tersebut wajib mengeluarkan kartu 3 diamond. Pemain dapat mengeluarkan 2, 3, atau bahkan 5 kartu sekaligus selama dari kombinasi kartu tersebut terdapat 3 diamond. Setelah gilirannya terlewati pemain selanjutnya harus mengeluarkan kartu yang lebih tinggi daripada kartu yang dikeluarkan oleh pemain sebelumnya. Pemain harus mengikuti paket yang berlangsung dalam putaran itu. Jika pemain tidak

dapat melawan kartu pemain sebelumnya maka dia harus *pass*. Jika semua pemain lain sudah *pass* maka pemain yang bermain terakhir dapat memulai putaran baru dengan bebas mengeluarkan kartu apapun. Pemain yang menghabiskan kartu di tangan terlebih dahulu adalah pemenangnya.



Gambar 3.2 Tampilan Game

3.1.3. Analisis Komponen Permainan

Ada beberapa komponen yang ada dalam aplikasi permainan 'Capsa Banting'. Komponen tersebut akan mempengaruhi jalannya permainan.

3.1.3.1. Pembagian Kartu

Pada awal permainan Capsa Banting pemain akan dibagikan kartu, dimana masing-masing pemain akan mendapat 13 kartu. Kartu yang dibagikan kepada setiap pemain bersifat pure random. Sifat pembagian kartu ini tidak berubah dan tidak bergantung pada komponen permainan lain. Oleh sebab itu permainan ini bersifat adil karena setiap pemain memiliki kesempatan menang yang sama.

3.1.3.2. Kartu

Kartu adalah objek dalam permainan yang nilainya akan dibandingkan untuk mengetahui apakah pemain dapat bermain atau tidak. Digunakan 52 kartu dalam permainan ini dimana kartu dibagikan dalam 4 kategori yaitu, diurutkan dari terkecil hingga terbesar, *Diamond*, *Club*, *Hearts*, *Spade*. Setiap kategori memiliki 13 kartu yang terdiri dari 9 kartu angka (angka 2 – 10) dan 4 kartu gambar (*Jack*, *Queen*, *King*, dan *As*).

3.1.3.3. Field

Field adalah tempat peletakan kartu yang sudah dimainkan oleh setiap pemain. Kartu yang paling terakhir dimainkan diletakkan di paling atas pada field. Pemain yang akan memainkan kartu harus melihat kartu terakhir di field untuk memilah kartu mana yang bisa pemain keluarkan.

3.1.3.4. Computer Player

Computer Player ikut bermain pada mode *single player* untuk menjadi lawan pemain. *Computer Player* juga bermain pada mode *multiplayer* jika pemain yang bergabung kurang dari 4 pemain. *Computer Player* inilah yang menerapkan algoritma Greedy dalam pemilihan kartunya.*Computer Player* terdiri dari 3 tingkat kesulitan yaitu *easy*, *normal*, *hard*.

3.2. Perancangan

3.2.1. Perancangan Free

Free adalah saat dimana pemain dapat mengeluarkan kartu secara bebas. Kondisi *Free* ini terjadi jika pemain berhasil memenangkan putaran sebelumnya dan berhak memulai putaran berikutnya. Kondisi *Free* ini juga dapat terjadi jika pemain memiliki kartu 3 diamond dan berhak memulai permainan. Pada kondisi *Free* ini *Computer Player* mengeluarkan kartu tergantung pada tingkat kesulitan yang berjalan.

3.2.1.1. Easy

Untuk tingkat kesulitan *easy*, *Computer Player* menerapkan pemilihan kartu secara random. Jadi kartu yang akan dikeluarkan hanyalah 1 kartu yang di pilih secara acak dari seluruh kartu yang ada di tangan. Cara ini kurang optimal karena kartu yang dipilih belum tertentu yang terkecil yang dapat dikeluarkan pada saat itu.

```
if(GameObject.Find("DifficultyScript").GetComponent<Di
fficulty>().difficulty == "Easy")
{
     cardsToPlay.Add(handCards[Random.Range(0,
handCards.Count)]);
     toField();
     Handler.cardPacket = "Single";
}
```

3.2.1.2. Medium

Untuk tingkat kesulitan *medium*, *Computer Player* menerapkan pemilihan kartu menggunakan algoritma Greedy, tidak lagi memilih kartu secara random. Kartu yang dipilih untuk dimainkan hanya berjumlah 1, sama seperti tingkat kesulitan *easy*, perbedaannya adalah pemilihan kartu ini tidak secara random lagi melainkan menggunakan algoritma Greedy. Logika pemilihan Greedy adalah dengan memilih kartu terendah yang mungkin dimainkan pada saat itu, dan karena giliran bebas maka kartu terbaik yang dikeluarkan adalah kartu terendah yang dimiliki di tangan. Pseudocode pemilihan kartu pada tingkat kesulitan *medium* adalah sebagai berikut :

```
if(GameObject.Find("DifficultyScript").GetComponent<Di
fficulty>().difficulty == "Medium")
{
     cardsToPlay.Add(handCards[0]);
     toField();
     Handler.cardPacket = "Single";
}
```

Kartu yang dipilih untuk dikeluarkan adalah kartu pertama ditangan karena sudah terjadi proses *sort* untuk mempermudah pemilihan kartu.

3.2.1.3. Hard

Untuk tingkat kesulitan *hard* tetap digunakan metode Greedy untuk pemilihan kartunya, tetapi berbeda dengan tingkat kesulitan *medium* kartu yang dipilih bisa lebih dari 1 kartu (*packet).*Sesuai dengan kunci dari memenangkan permainan ini yaitu menghabiskan kartu di tangan secepat mungkin, maka pengecekan kartu dimulai paket dimana jumlah kartu merupakan jumlah kartu terbanyak yang bisa dikeluarkan dalam 1 giliran, yaitu 5 kartu. Pengecekan dimulai dari paket 5 kartu yang terendah hingga yang tertinggi (*straight*, *flush*, *fullhouse*, *four of a kind*, *straight flush*, dan *royal straight flush*). Jika tidak ditemukan maka akan dipilih kombinasi kartu dengan jumlah kartu terbanyak ke dua yaitu 3 kartu (*threes*), dilanjutkan dengan 2 kartu (*pair*) dan yang terakhir 1 kartu (*singles*). Dalam pemilihan kartu untuk setiap paket juga dilakukan pemilihan secara Greedy dengan metode yang sama dengan pemilihan kartu *Computer Player non-free*, yaitu dengan memilih 1 kartu terendah sebagai acuan dan mencari kombinasi yang mungkin dari kartu tersebut. Pseudocode pemilihan kartu pada tingkat kesulitan *hard* adalah sebagai berikut :

```
Elseif(GameObject.Find("DifficultyScript").GetComponent<Difficulty
>().difficulty == "Hard")
{
        if (findStraight())
        {
                toField();
                Handler.cardPacket = "Straight";
        }
        else if (findFlush())
        {
                toField();
                Handler.cardPacket = "Flush";
        }
        else if (findFullHouse())
```

```
        {
                toField();
                Handler.cardPacket = "FullHouse";
        }
        else if (findFourOfAKind())
        {
                toField();
                Handler.cardPacket = "FourOfAKind";
        }
        else if (findStraightFlush())
        {
                toField();
                Handler.cardPacket = "StraightFlush";
        }
        else if (findRoyalStraightFlush())
        {
                toField();
                Handler.cardPacket = "RoyalStraightFlush";
        }
        else if (findThrees())
        {
                toField();
                Handler.cardPacket = "Threes";
        }
        else if (findPair())
        {
                toField();
                Handler.cardPacket = "Pair";
        }
        else
        {
                cardsToPlay.Add(handCards[0]);
                toField();
                Handler.cardPacket = "Single";
        }
}
```

3.2.2. Perancangan Non-Free

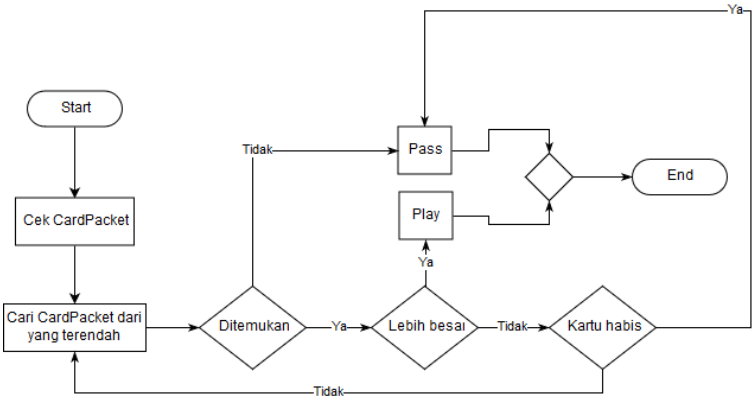
Non-Free adalah kondisi dimana pemain harus menimpa kartu yang sudah ada di *field* pada saat itu. Kondisi Non-Free ini terjadi ketika pemain lain sudah mengeluarkan kartu pada giliran sebelumnya sehingga pemain selanjutnya harus mengeluarkan kartu yang bernilai lebih besar daripada kartu yang sudah dikeluarkan. Pada kondisi inilah *Computer Player* akan menggunakan algoritma Greedy untuk mencara kartu paling

optimal yang akan dikeluarkan untuk melawan kartu yang ada pada *field*.

3.2.3. Perancangan Algoritma Greedy

Algoritma Greedy diterapkan pada *gameplay* game Capsa Banting pada bagian pemilihan kartu yang akan dikeluarkan. Algoritma Greedy yang diterapkan memiliki beberapa perubahan. Meskipun terjadi beberapa perubahan, inti dari algoritmanya sendiri tidak berubah. Beberapa perubahan tersebut dirubah agar dapat menyesuaikan dengan *gameplay* Capsa Banting.

Penyesuaian terhadap algoritma Greedy terdapat pada kondisi saat proses pemilihan kartu. Greedy yang digunakan sebagai pemilihan kartu pada game Capsa Banting memiliki bagian yang sama dengan algoritma Greedy biasa. Gambaran dari bagaimana jalannya pemilihan kartu dengan algoritma greedydapat dilihat di gambar 3.2.



Gambar 3.3 Flow diagram Algoritma Greedy (Non-Free)

Berdasarkan gambar tersebut, berikut adalah tahapan perancangan dan penerapan algoritma Greedy pada aplikasi permainan untuk kasus dimana *Computer Player* harus menimpa kartu yang sudah ada di *field*:

1. Cek CardPacket yang terakhir dimainkan agar dapat menentukan kombinasi kartu mana yang dapat digunakan pada putaran tersebut.
2. Setelah diketahui CardPacket yang sedang berlangsung di putaran tersebut maka algoritma greedy mulai mencari kartu-kartu yang bisa disusun menjadi CardPacket sekarang, dimulai dengan mencari kartu terendah yang mungkin dikeluarkan sebagai kartu acuan. Jika sudah didapat maka akan dicari kartu selanjutnya yang diperlukan untuk melengkapi kombinasi tersebut dimulai dari yang paling kecil.
3. Jika kombinasi kartu CardPacket tidak ditemukan maka *Computer Player* akan melewatkan giliran (*Pass*). *Computer Player* yang melewatkan giliran harus menunggu ronde selanjutnya agar dapat mengeluarkan kartu lagi.
4. Jika ditemukan maka akan dilakukan pengecekan kedua yaitu apakah CardPacket *Computer Player* nilainya lebih tinggi dari pada CardPacket yang ada di *field*. Jika ya maka CardPacket tersebut akan dimainkan dan dilanjutkan oleh pemain giliran selanjutnya.
5. Jika CardPacket yang akan dimainkan ternyata bernilai lebih rendah daripada CardPacket yang ada di *field* maka akan dilakukan pengecekan kembali yaitu apakah sudah semua kombinasi dicoba. Kombinasi yang dicoba adalah semua kombinasi yang mungkin dibuat menggunakan kartu acuan saat itu.
6. Jika masih ada kartu yang belum dipakai sebagai acuan maka kembali ke poin 2 yaitu pencarian CardPacket selanjutnya. Perbedaanya adalah kartu acuan yang dipilih sebagai dasar penyusunan kombinasi kartu bukan kartu terendah lagi melainkan

kartu kedua terendah, tergantung pada perulangan keberapa.

7. Jika sudah tidak ada lagi maka *Computer Player* harus melewatkan giliran.

Berbeda dengan Greedy biasa, pada Greedy yang digunakan dalam 'Capsa Banting' bisa jadi ada kartu yang tidak dicek. Kartu yang tidak di cek tersebut menandakan bahwa kartu tersebut tidak lebih besar daripada kartu yang sudah ada di *field* sekarang. Hal tersebut dikarenakan adanya persyaratan kartu mana yang bisa dikeluarkan. Namun, titik yang tidak di cek tersebut akan dianggap sebagai objek jalan biasa saat dilakukan realisasi terhadap tampilan *gameplay*.

Flow diagram diatas adalah untuk melawan kartu yang sudah ada di field. Jika *Computer Player* mendapat giliran bebas, maka akan digunakan algoritma Greedy yang sedikit berbeda. Tidak semua tingkat kesulitan *Computer Player* menggunakan algoiritma Greedy. Algortima Greedy hanya digunakan pada *Computer Player* dengan tingkat kesulitan *medium* dan *hard*.

3.2.4. Perancangan Asset

Semua asset gambar merupakan asset internal penulis yang dibuat oleh penulis sendiri. Asset suara didapatkan oleh penulis dari audioblocks.com melalui akun Lab IGS.

3.2.5. Perancangan Mode

Terdapat 2 mode permainan pada game "Capsa Banting" yaitu mode *singleplayer* dan *multiplayer*. Pada mode *singleplayer* hanya terdapat 1 pemain (*Human Player)* sedangkan 3 pemain lain merupakan *Computer Player*. Pada mode *multiplayer* pemain dapat bermain bersama pemain lain. Jika jumlah pemain kurang dari 4 maka akan di masukkan *Computer Player* sejumlah pemain yang kurang. Pada mode *multiplayer* sendiri terdapat 3 menu yaitu *dedicated server*, dimana *device* yang digunakan akan menjadi *server* tersendiri dan menangani semua kegiatan *client*, *play and host*, dimana *device* berperan sebagai *host* sekaligus *client*, dan *join*, dimana *device* berperan sebagai *client* yang akan bergabung dengan *host*.

3.2.6. Perancangan Realisasi Tampilan Permainan

Untuk tampilan permainan terdapat beberapa objek baik *interactable* dan *non-interactable*bagi pemain.Terdapat 3 objek *interactable* utama yang antara lain adalah *Options UI*, yang terdiri dari tompol *pause* dan *exit*, *Gameplay UI*, yang terdiri dari tombol *play* dan *pause*, dan yang terakhir adalah *hand cards* yang merupakan kartu yang dimiliki oleh pemain yang bersangkutan. Sementara untuk objek *non-interactable* terdiri dari, *Status UI*, yang berisi data-data game seperti jumlah kartu yang sudah dimainkan, putaran, giliran, serta *card packet* yang sedang dimainkan sekarang, *Field Cards*, yaitu tumpukan kartu yang sudah dimainkan pada putaran tersebut, dan yang terakhir adalah *hand cards* pemain lain.

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI SISTEM

Bab ini membahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Implementasi yang dijelaskan adalah bagaimana menerapkan algoritma Greedy dalam game Capsa Banting untuk *Computer Player* yang kompetitif. Game Engine yang digunakan adalah Unity dengan bahasa pemrograman C#.

4.1. Lingkungan Pengembangan Sistem

Lingkungan pengembangan sistem yang digunakan untuk mengembangkan Tugas Akhir ini dilakukan pada lingkungan pengembangan sebagai berikut.

1. Sistem operasi Windows 10 Education 64 bit.
2. Unity Game Engine 5.6.5.f1 Personal
3. Visual Studio 2017 sebagai editor

4.2. Implementasi *Gameplay*

Implementasi dari semua fungsi diwujudkan dalam bentuk code dengan Bahasa pemrograman C#.

4.2.1. Implementasi Aksi Pemain

Pada saat pemilihan kartu oleh pemain, pemain dapat menentukan kartu apa saja yang mau dimainkan. Pemain bisa memilih dan mengganti kartu yang sudah dipilih sampai mendapatkan kombinasi kartu yang diinginkan. sebagaimana yang telah dijelaskan dapat dilihat pada Kode Sumber 4.1.

```
1.  void OnMouseDown() {
2.      if (this.transform.parent.name == "P1" && GameObject.Find("P1").Get
    Component < Player > ().played == false) {
3.          cardClicked.Play();
4.          if (clicked) {
5.              this.transform.position = new Vector3(this.transform.positi
    on.x, this.transform.position.y - 0.5 f, this.transform.position.z);
6.              clicked = false;
```

```
7.              (GameObject.Find("P1")).GetComponent < Player > ().cardsToP
    lay.Remove(showCard);
8.          } else {
9.              this.transform.position = new Vector3(this.transform.positi
    on.x, this.transform.position.y + 0.5 f, this.transform.position.z);
10.             clicked = true;
11.             (GameObject.Find("P1")).GetComponent < Player > ().cardsToP
    lay.Add(showCard);
12.         }
13.     }
14. }
```

Kode Sumber 4.1 Implementasi Aksi *Click* Pemain pada Kartu

4.2.2. Implementasi Pengecekan Kartu

Di saat pemain mengeluarkan kartu maka akan dilakukan pengecekan terhadap kartu-kartu yang telah dipiliholeh pemain. Apakah kartu tersebut bisa digunakan di dalam putaran tersebut atau tidak. Jika bisa maka kartu akan di keluarkan, jika tidak maka tidak dilakukan aksi apapun.Kode dapat dilihat di Kode Sumber 4.2.

```
1.  public void cardChecker() {
2.      if (Handler.turn == 0 && played == false) {
3.          if (!pass) {
4.              CheckerObject.GetComponent < Checker > ().cardsToBeChecked.
    Clear();
5.              for (int i = 0; i < cardsToPlay.Count; i++) {
6.                  CheckerObject.GetComponent < Checker > ().cardsToBeChec
    ked.Add(cardsToPlay[i]);
7.              }
8.              if (Handler.cardPacket == "Free") {
9.                  if (CheckerObject.GetComponent < Checker > ().isSingle(
    )) {
10.                     if (Handler.gameStart) {
11.                         toField();
12.                         Handler.cardPacket = "Single";
13.                     } else {
14.                         if (cardsToPlay.Contains(0)) {
15.                             Handler.gameStart = true;
16.                             toField();
```

```
                            Handler.cardPacket = "Single";
                        }
                    }
                } else if (CheckerObject.GetComponent < Checker > ().is
Pair()) {
                    if (Handler.gameStart) {
                        toField();
                        Handler.cardPacket = "Pair";
                    } else {
                        if (cardsToPlay.Contains(0)) {
                            Handler.gameStart = true;
                            toField();
                            Handler.cardPacket = "Pair";
                        }
                    }
                } else if (CheckerObject.GetComponent < Checker > ().is
Threes()) {
                    if (Handler.gameStart) {
                        toField();
                        Handler.cardPacket = "Threes";
                    } else {
                        if (cardsToPlay.Contains(0)) {
                            Handler.gameStart = true;
                            toField();
                            Handler.cardPacket = "Threes";
                        }
                    }
                } else if (CheckerObject.GetComponent < Checker > ().is
Straight()) {
                    if (CheckerObject.GetComponent < Checker > ().isFlu
sh()) {
                        if (CheckerObject.GetComponent < Checker > ().i
sRoyalStraightFlush()) {
                            Handler.cardPacket = "RoyalStraightFlush";

                        } else {
                            Handler.cardPacket = "StraightFlush";
                        }
                    } else {
                        Handler.cardPacket = "Straight";
                    }
                    toField();
```

```
53.                 } else if (CheckerObject.GetComponent < Checker > ().is
    Flush()) {
54.                     toField();
55.                     Handler.cardPacket = "Flush";
56.                 } else if (CheckerObject.GetComponent < Checker > ().is
    FullHouse()) {
57.                     toField();
58.                     Handler.cardPacket = "FullHouse";
59.                 } else if (CheckerObject.GetComponent < Checker > ().is
    FourOfAKind()) {
60.                     toField();
61.                     Handler.cardPacket = "FourOfAKind";
62.                 }
63.             } else if (Handler.cardPacket == "Single" && CheckerObject.
    GetComponent < Checker > ().isSingle()) {
64.                 if (cardsToPlay[0] > HandlerObject.GetComponent < Handl
    er > ().fieldCards[0]) {
65.                     toField();
66.                 }
67.             } else if (Handler.cardPacket == "Pair" && CheckerObject.Ge
    tComponent < Checker > ().isPair()) {
68.                 if (Mathf.Max(cardsToPlay.ToArray()) > Mathf.Max(Handle
    rObject.GetComponent < Handler > ().fieldCards.ToArray())) {
69.                     toField();
70.                 }
71.             } else if (Handler.cardPacket == "Threes" && CheckerObject.
    GetComponent < Checker > ().isThrees()) {
72.                 if (Mathf.Max(cardsToPlay.ToArray()) > Mathf.Max(Handle
    rObject.GetComponent < Handler > ().fieldCards.ToArray())) {
73.                     toField();
74.                 }
75.             } else {
76.                 if (Handler.cardPacket == "Straight") {
77.                     if (CheckerObject.GetComponent < Checker > ().isStr
    aight()) {
78.                         if (CheckerObject.GetComponent < Checker > ().i
    sFlush()) {
79.                             if (CheckerObject.GetComponent < Checker >
    ().isRoyalStraightFlush()) {
80.                                 Handler.cardPacket = "RoyalStraightFlus
    h";
81.                                 toField();
82.                             } else {
```

```
83.                                 Handler.cardPacket = "StraightFlush";
84.                                 toField();
85.                             }
86.                         } else {
87.                             if (Mathf.Max(cardsToPlay.ToArray()) > Math
    f.Max(HandlerObject.GetComponent < Handler > ().fieldCards.ToArray()))
    {
88.                                 toField();
89.                                 Handler.cardPacket = "Straight";
90.                             }
91.                         }
92.                     } else if (CheckerObject.GetComponent < Checker > (
    ).isFlush()) {
93.                         toField();
94.                         Handler.cardPacket = "Flush";
95.                     } else if (CheckerObject.GetComponent < Checker > (
    ).isFullHouse()) {
96.                         toField();
97.                         Handler.cardPacket = "FullHouse";
98.                     } else if (CheckerObject.GetComponent < Checker > (
    ).isFourOfAKind()) {
99.                         toField();
100.                             Handler.cardPacket = "FourOfAKind";
101.                         }
102.                     } else if (Handler.cardPacket == "Flush") {
103.                         if (CheckerObject.GetComponent < Checker > ().
    isFlush()) {
104.                             if (cardsToPlay[0] % 4 > HandlerObject.Get
    Component < Handler > ().fieldCards[0] % 4) {
105.                                 toField();
106.                                 Handler.cardPacket = "Flush";
107.                             } else if (cardsToPlay[0] % 4 == HandlerOb
    ject.GetComponent < Handler > ().fieldCards[0] % 4) {
108.                                 if (Mathf.Max(cardsToPlay.ToArray()) >
     Mathf.Max(HandlerObject.GetComponent < Handler > ().fieldCards.ToArray
    ())) {
109.                                     toField();
110.                                     Handler.cardPacket = "Flush";
111.                                 }
112.                             }
113.                         } else if (CheckerObject.GetComponent < Checke
    r > ().isFullHouse()) {
114.                             toField();
```

```
115.                                 Handler.cardPacket = "FullHouse";
116.                             } else if (CheckerObject.GetComponent < Checke
    r > ().isFourOfAKind()) {
117.                                 toField();
118.                                 Handler.cardPacket = "FourOfAKind";
119.                             } else if (CheckerObject.GetComponent < Checke
    r > ().isStraightFlush()) {
120.                                 if (CheckerObject.GetComponent < Checker >
     ().isRoyalStraightFlush()) {
121.                                     toField();
122.                                     Handler.cardPacket = "RoyalStraightFlu
    sh";
123.                                 } else {
124.                                     toField();
125.                                     Handler.cardPacket = "StraightFlush";

126.                                 }
127.                             }
128.                         } else if (Handler.cardPacket == "FullHouse") {
129.                             cardsToPlay.Sort();
130.                             if (CheckerObject.GetComponent < Checker > ().
    isFullHouse()) {
131.                                 if (cardsToPlay[1] / 4 == cardsToPlay[2] /
     4) {
132.                                     if (HandlerObject.GetComponent < Handl
    er > ().fieldCards[1] / 4 == HandlerObject.GetComponent < Handler > ().
    fieldCards[2] / 4) {
133.                                         if (cardsToPlay[0] > HandlerObject
    .GetComponent < Handler > ().fieldCards[0]) {
134.                                             toField();
135.                                             Handler.cardPacket = "FullHous
    e";
136.                                         }
137.                                     } else {
138.                                         if (cardsToPlay[0] > HandlerObject
    .GetComponent < Handler > ().fieldCards[4]) {
139.                                             toField();
140.                                             Handler.cardPacket = "FullHous
    e";
141.                                         }
142.                                     }
143.                                 } else {
```

```
144.                                 if (HandlerObject.GetComponent < Handl
    er > ().fieldCards[1] / 4 == HandlerObject.GetComponent < Handler > ().
    fieldCards[2] / 4) {
145.                                     if (cardsToPlay[4] > HandlerObject
    .GetComponent < Handler > ().fieldCards[0]) {
146.                                         toField();
147.                                         Handler.cardPacket = "FullHous
    e";
148.                                     }
149.                                 } else {
150.                                     if (cardsToPlay[4] > HandlerObject
    .GetComponent < Handler > ().fieldCards[4]) {
151.                                         toField();
152.                                         Handler.cardPacket = "FullHous
    e";
153.                                     }
154.                                 }
155.                             }
156.                         } else if (CheckerObject.GetComponent < Checke
    r > ().isFourOfAKind()) {
157.                             toField();
158.                             Handler.cardPacket = "FourOfAKind";
159.                         } else if (CheckerObject.GetComponent < Checke
    r > ().isStraightFlush()) {
160.                             if (CheckerObject.GetComponent < Checker >
     ().isRoyalStraightFlush()) {
161.                                 toField();
162.                                 Handler.cardPacket = "RoyalStraightFlu
    sh";
163.                             } else {
164.                                 toField();
165.                                 Handler.cardPacket = "StraightFlush";

166.                             }
167.                         }
168.                     } else if (Handler.cardPacket == "FourOfAKind") {

169.                         if (CheckerObject.GetComponent < Checker > ().
    isFourOfAKind()) {
170.                             if (cardsToPlay[0] / 4 == cardsToPlay[1] /
     4) {
```

```
171.                                 if (HandlerObject.GetComponent < Handl
    er > ().fieldCards[0] / 4 == HandlerObject.GetComponent < Handler > ().
    fieldCards[1] / 4) {
172.                                     if (cardsToPlay[0] > HandlerObject
    .GetComponent < Handler > ().fieldCards[0]) {
173.                                         toField();
174.                                         Handler.cardPacket = "FourOfAK
    ind";
175.                                     }
176.                                 } else {
177.                                     if (cardsToPlay[0] > HandlerObject
    .GetComponent < Handler > ().fieldCards[4]) {
178.                                         toField();
179.                                         Handler.cardPacket = "FourOfAK
    ind";
180.                                     }
181.                                 }
182.                             } else {
183.                                 if (HandlerObject.GetComponent < Handl
    er > ().fieldCards[0] / 4 == HandlerObject.GetComponent < Handler > ().
    fieldCards[1] / 4) {
184.                                     if (cardsToPlay[4] > HandlerObject
    .GetComponent < Handler > ().fieldCards[0]) {
185.                                         toField();
186.                                         Handler.cardPacket = "FourOfAK
    ind";
187.                                     }
188.                                 } else {
189.                                     if (cardsToPlay[4] > HandlerObject
    .GetComponent < Handler > ().fieldCards[4]) {
190.                                         toField();
191.                                         Handler.cardPacket = "FourOfAK
    ind";
192.                                     }
193.                                 }
194.                             }
195.                         } else if (CheckerObject.GetComponent < Checke
    r > ().isStraightFlush()) {
196.                             if (CheckerObject.GetComponent < Checker >
     ().isRoyalStraightFlush()) {
197.                                 toField();
198.                                 Handler.cardPacket = "RoyalStraightFlu
    sh";
```

```
199.                                } else {
200.                                    toField();
201.                                    Handler.cardPacket = "StraightFlush";

202.                                }
203.                            }
204.                        } else if (Handler.cardPacket == "StraightFlush")
    {
205.                            if (CheckerObject.GetComponent < Checker > ().
    isStraightFlush()) {
206.                                if (CheckerObject.GetComponent < Checker >
     ().isRoyalStraightFlush()) {
207.                                    toField();
208.                                    Handler.cardPacket = "RoyalStraightFlu
    sh";
209.                                } else {
210.                                    toField();
211.                                    Handler.cardPacket = "StraightFlush";
212.                                }
213.                            }
214.                        } else if (Handler.cardPacket == "RoyalStraightFlu
    sh") {
215.                            passTurn();
216.                        }
217.                    }
218.                }
219.            }
220.        }
```

Kode Sumber 4.2 Pengecekan Kartu

4.2.3. Implementasi *Play*

Kartu hanya dapat di letakkan di *field* permainan jika memenuhi kondisi, yaitu memiliki paket yang sesuai dengan putaran yang dimainkan dan bernilai lebih besar dari kartu terakhir yang di *field*. Tetapi apabila pemain mendapat giliran bebas maka kartu akan diletakkan jika sesuai dengan aturan kombinasi kartu yang ada. Kode dapat dilihat di Kode Sumber 4.3.

```
1.  public void toField() {
2.      cardSpin.Play();
```

```
3.      playOrPause.Play();
4.      StartCoroutine(CardsAnimated());
5.      played = true;
6.      Handler.lastPlay = Handler.turn;
7.  }
8.  IEnumerator CardsAnimated() {
9.      HandlerObject.GetComponent < Handler > ().fieldCards.Clear();
10.     cardsToPlay.Sort();
11.     for (int x = 0; x < cardsToPlay.Count; x++) {
12.         foreach(Transform child in transform) {
13.             if (child.GetComponent < CardModel > ().showCard == cardsTo
    Play[x]) {
14.                 child.GetComponent < CardModel > ().TriggerDestroy();
15.                 break;
16.             }
17.         }
18.     }
19.     Instantiate(spinUp, gameObject.transform.position, Quaternion.ident
    ity);
20.     yield
21.     return new WaitForSeconds(0.5 f);
22.     for (int x = 0; x < cardsToPlay.Count; x++) {
23.         Handler.cardsOnField++;
24.         GameObject instance = Instantiate(cardPlayed, new Vector3(0.5 f
     * x - (cardsToPlay.Count * 0.25 f) + 0.25 f, 0, -
    0.1 f * Handler.cardsOnField), Quaternion.identity);
25.         instance.GetComponent < CardModel > ().ToggleCard(cardsToPlay[x
    ]);
26.         instance.tag = "CARDSONTHEFIELD";
27.         handCards.Remove(cardsToPlay[x]);
28.     }
29.     for (int i = 0; i < cardsToPlay.Count; i++) {
30.         HandlerObject.GetComponent < Handler > ().fieldCards.Add(cardsT
    oPlay[i]);
31.         msg += ", " + cardsToPlay[i];
32.     }
33.     Debug.Log("P1 Played " + msg);
34.     msg = "";
35.     cardsToPlay.Clear();
36.     yield
37.     return new WaitForSeconds(1.0 f);
38.     if (handCards.Count == 0) {
39.         win();
```

```
    }
    Handler.turn++;
    played = false;
}
```

Kode Sumber 4.3 Peletakan Kartu

4.2.4. Implementasi *Pass*

Melewatkan giliran atau *pass* dilakukan jika pemain tidak bisa atau tidak mau melawan kartu terakhir yang sudah ada *field*. Dalam kasus ini maka pemain tidak bisa lagi mengeluarkan kartu hingga ronde tersebut selesai (3 pemain melewatkan giliran). Kode dapat dilihat di Kode Sumber 4.4.

```
public void passTurn() {
    if (!pass && Handler.cardPacket != "Free" && Handler.turn == 0) {
        playOrPause.Play();
        Debug.Log("P1 Passed");
        pass = true;
        HandlerObject.GetComponent < Handler > ().playerPassed.Add("P1"
);
        if (HandlerObject.GetComponent < Handler > ().playerPassed.Coun
t >= 3) {
            HandlerObject.GetComponent < Handler > ().NewRound();
        } else {
            Handler.turn++;
        }
    }
}
```

Kode Sumber 4.4 Melewatkan Giliran

4.2.5. Implementasi *OptionsUI*

OptionsUI yang dimaksud adalah tombol *pause* yang dapat menghentikan permainan sementara dan tombol *exit* untuk keluar dari permainan. Kode dapat dilihat di Kode Sumber 4.5.

```
public void Pause() {
    Time.timeScale = 0;
    pauseUI.SetActive(true);
    GameObject.Find("Pause/Exit").transform.GetChild(0).gameObject.SetA
ctive(false);
```

```
5.      GameObject.Find("Pause/Exit").transform.GetChild(1).gameObject.SetA
    ctive(false);
6.  }
7.  public void Resume() {
8.      Time.timeScale = 1;
9.      pauseUI.SetActive(false);
10.     exitUI.SetActive(false);
11.     GameObject.Find("Pause/Exit").transform.GetChild(0).gameObject.SetA
    ctive(true);
12.     GameObject.Find("Pause/Exit").transform.GetChild(1).gameObject.SetA
    ctive(true);
13. }
14. public void Exit() {
15.     Time.timeScale = 0;
16.     exitUI.SetActive(true);
17.     GameObject.Find("Pause/Exit").transform.GetChild(0).gameObject.SetA
    ctive(false);
18.     GameObject.Find("Pause/Exit").transform.GetChild(1).gameObject.SetA
    ctive(false);
19. }
20. public void ExitYes() {
21.     SceneManager.LoadScene("Menu");
22. }
```

Kode Sumber 4.5 *Options UI*

4.2.6. Implementasi Stage

Pengaturan stage yang mencakup segala hal yang berhubungan dengan pengaturan *load stage* dalam permainan. Kode dapat dilihat di Kode Sumber 4.6.

```
1.  public void Singleplayer() {
2.      difficultyPopUp.SetActive(true);
3.      foreach(Transform child in transform) {
4.          child.gameObject.SetActive(false);
5.      }
6.  }
7.  public void Easy() {
8.      GameObject.Find("DifficultyScript").GetComponent < Difficulty > ().
    difficulty = "Easy";
9.      SceneManager.LoadScene("Game");
10. }
```

```
11. public void Medium() {
12.     GameObject.Find("DifficultyScript").GetComponent < Difficulty > ().
    difficulty = "Medium";
13.     SceneManager.LoadScene("Game");
14. }
15. public void Hard() {
16.     GameObject.Find("DifficultyScript").GetComponent < Difficulty > ().
    difficulty = "Hard";
17.     SceneManager.LoadScene("Game");
18. }
19. public void Multiplayer() {
20.     SceneManager.LoadScene("MPLobby");
21. }
22. public void Exit() {
23.     Application.Quit();
24. }
```

Kode Sumber 4.6 Pemilihan Stage

4.2.7. Implementasi Multiplayer

Sebagian besar bagian dari *multiplayer* mirip dengan *singleplayer*. Perbedaannya adalah penanganan *variable* dan animasi harus di ubah menjadi hubungan *server* dan *client*, dimana semua penyimpanan status dan data seperti, jumlah kartu, pengacakan kartu dan pembagiannya, serta kartu-kartu yang sudah ada di *field*.

4.3. Implementasi Antarmuka

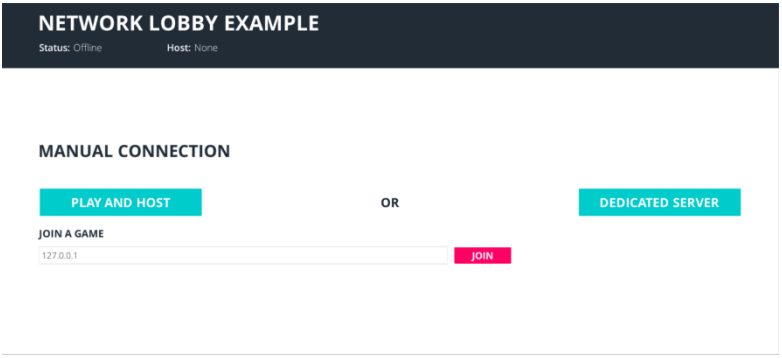
4.3.1. Implementasi *Main Menu*

Implementasi dari *Main Menu* mencakup 3 pilihan. Single Player dimana pemain bermain sendiri melawan 3 *Computer Player*, *Multiplayer* jika ingin bermain bersama pemain lain secara *real-time*, dan *Exit* untuk keluar dari permainan. *Screenshot* dari Main Menu dapat dilihat pada Gambar 4.1.

Gambar 4.1 Implementasi tampilan *Main Menu*

4.3.2. Implementasi Multiplayer Lobby

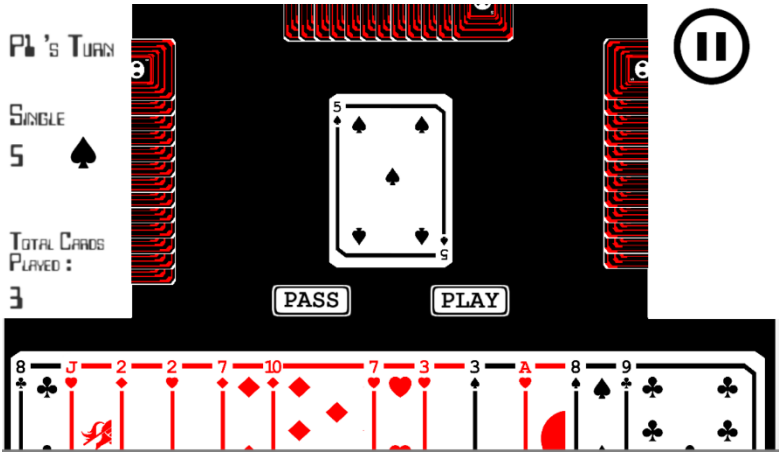
Implementasi *Multiplayer Lobby* mencakup opsi *play and host* dimana *device* menjadi *client* sekaligus *server*. Terdapat juga opsi *dedicated server* dimana satu *device* dikhususkun untuk menangani kegiatan *server*. Terdapat juga opsi untuk bergabung pada room yang sudah dibuat dengan cara join ke IP pembuat nya.



Gambar 4.2 Implementasi Tampilan *Multiplayer Lobby*

4.3.3. Implementasi Gameplay Stage

Implementasi *Gameplay Stage* mencakup komponen stage berupa kartu kartu yang ada di tangan setiap pemain, tombol *play* dan *pause*, kartu-kartu yang berada di *field*, *sidebar* kiri yang berisi status seperti giliran, *CardPacket*, total kartu yang sudah dimainkan, dan *sidebar* kanan yang berisi opsi *pause* dan *exit*.*Screenshot* dari *gameplay* dapat dilihat pada Gambar 4.3.



Gambar 4.3 Implementasi Gameplay Stage

4.4. Implementasi Greedy

Pada bagian ini menjelaskan bagaimana menerapkan aturan algoritma Greedy untuk digunakan ke aplikasi permainan dalam pemilihan kartu *Computer Player*. Penerapan algoritma Greedy akan berjalan berbeda-beda untuk setiap *CardPacket*yang ada, kecuali pada *CardPacket Royal Straight Flush* dikarenakan kombinasi kartu yang bisa membentuk *Royal Straight Flush* hanya satu.

4.4.1. Single

```
1.  else if (Handler.cardPacket == "Single") {
2.      bool play = false;
3.      for (int x = 0; x < handCards.Count; x++) {
```

```
4.          if (handCards[x] > HandlerObject.GetComponent < Handler > ().fi
    eldCards[0]) {
5.              play = true;
6.              cardsToPlay.Add(handCards[x]);
7.              toField();
8.              break;
9.          }
10.     }
11.     if (!play) {
12.         passTurn();
13.     }
14. }
```

Kode Sumber 4.7 Greedy untuk *CardPacket Single*

4.4.2. Pair

```
1.  public bool findPair() {
2.      for (int x = 1; x < handCards.Count; x++) {
3.          if (handCards[x -
     1] / 4 == handCards[x] / 4 && handCards[x] > Mathf.Max(HandlerObject.G
    etComponent < Handler > ().fieldCards.ToArray())) {
4.              cardsToPlay.Add(handCards[x - 1]);
5.              cardsToPlay.Add(handCards[x]);
6.              return true;
7.          }
8.      }
9.      return false;
10. }
```

Kode Sumber 4.8 Greedy untuk *CardPacket Pair*

4.4.3. Threes

```
1.  public bool findThrees() {
2.      for (int x = 2; x < handCards.Count; x++) {
3.          if (handCards[x - 2] / 4 == handCards[x] / 4 && handCards[x -
     1] / 4 == handCards[x] / 4 && handCards[x] > Mathf.Max(HandlerObject.G
    etComponent < Handler > ().fieldCards.ToArray())) {
4.              cardsToPlay.Add(handCards[x - 2]);
5.              cardsToPlay.Add(handCards[x - 1]);
6.              cardsToPlay.Add(handCards[x]);
7.              return true;
8.          }
9.      }
10.     return false;
```

```
11. }
```

Kode Sumber 4.9 Greedy untuk *CardPacket Threes*

4.4.4. Straight

```
1.  public bool findStraight() {
2.      if (Handler.cardPacket != "Straight") {
3.          return false;
4.      }
5.      for (int x = 0; x < handCards.Count - 4; x++) {
6.          cardsToPlay.Add(handCards[x]);
7.          for (int y = x + 1; y < handCards.Count; y++) {
8.              if (handCards[y] / 4 == Mathf.Max(cardsToPlay.ToArray()) /
    4 + 1) {
9.                  cardsToPlay.Add(handCards[y]);
10.             }
11.             if (cardsToPlay.Count == 5 || handCards[y] / 4 > Mathf.Max(
    cardsToPlay.ToArray()) / 4 + 1) {
12.                 break;
13.             }
14.         }
15.         if (cardsToPlay.Count == 5 && Mathf.Max(cardsToPlay.ToArray())
    > Mathf.Max(HandlerObject.GetComponent < Handler > ().fieldCards.ToArra
    y())) {
16.             bool temp = false;
17.             for (int z = 0; z < cardsToPlay.Count - 1; z++) {
18.                 if (cardsToPlay[z] % 4 != cardsToPlay[z + 1] % 4) {
19.                     temp = true;
20.                     break;
21.                 }
22.             }
23.             if (temp) {
24.                 return true;
25.             } else {
26.                 cardsToPlay.Clear();
27.             }
28.         } else {
29.             cardsToPlay.Clear();
30.         }
31.     }
32.     return false;
33. }
```

Kode Sumber 4.10 Greedy untuk *CardPacket Straight*

4.4.5. Flush

```
1.  public bool findFlush() {
2.      for (int x = 0; x < handCards.Count - 4; x++) {
3.          cardsToPlay.Add(handCards[x]);
4.          for (int y = x + 1; y < handCards.Count; y++) {
5.              if (handCards[y] % 4 == cardsToPlay[0] % 4) {
6.                  cardsToPlay.Add(handCards[y]);
7.              }
8.              if (cardsToPlay.Count == 5) {
9.                  break;
10.             }
11.         }
12.         if (cardsToPlay.Count == 5) {
13.             if (Handler.cardPacket != "Flush" || (Handler.cardPacket ==
     "Flush" && Mathf.Max(cardsToPlay.ToArray()) > Mathf.Max(HandlerObject.
    GetComponent < Handler > ().fieldCards.ToArray()))) {
14.                 bool temp = false;
15.                 for (int z = 0; z < cardsToPlay.Count - 1; z++) {
16.                     if ((cardsToPlay[z] / 4) + 1 != cardsToPlay[z + 1]
    / 4) {
17.                         temp = true;
18.                         break;
19.                     }
20.                 }
21.                 if (temp) {
22.                     return true;
23.                 } else {
24.                     cardsToPlay.Clear();
25.                 }
26.             } else {
27.                 cardsToPlay.Clear();
28.             }
29.         } else {
30.             cardsToPlay.Clear();
31.         }
32.     }
33.     return false;
34. }
```

Kode Sumber 4.11 Greedy untuk *CardPacket Flush*

4.4.6. Full House

```
1.  public bool findFullHouse() {
```

```
2.      for (int x = 0; x < handCards.Count - 2; x++) {
3.          if (handCards[x] / 4 == handCards[x + 1] / 4 && handCards[x] /
    4 == handCards[x + 2] / 4) {
4.              cardsToPlay.Add(handCards[x]);
5.              cardsToPlay.Add(handCards[x + 1]);
6.              cardsToPlay.Add(handCards[x + 2]);
7.              if (Handler.cardPacket == "FullHouse") {
8.                  if (HandlerObject.GetComponent < Handler > ().fieldCard
    s[1] / 4 == HandlerObject.GetComponent < Handler > ().fieldCards[2] / 4
    ) {
9.                      if (cardsToPlay[0] > HandlerObject.GetComponent < H
    andler > ().fieldCards[0]) {
10.                         break;
11.                     } else {
12.                         cardsToPlay.Clear();
13.                     }
14.                 } else {
15.                     if (cardsToPlay[0] > HandlerObject.GetComponent < H
    andler > ().fieldCards[4]) {
16.                         break;
17.                     } else {
18.                         cardsToPlay.Clear();
19.                     }
20.                 }
21.             } else {
22.                 break;
23.             }
24.         }
25.     }
26.     if (cardsToPlay.Count < 3) {
27.         return false;
28.     }
29.     for (int x = 0; x < handCards.Count - 1; x++) {
30.         if (handCards[x] / 4 == handCards[x + 1] / 4 && handCards[x] /
    4 != cardsToPlay[0] / 4) {
31.             cardsToPlay.Add(handCards[x]);
32.             cardsToPlay.Add(handCards[x + 1]);
33.             break;
34.         }
35.     }
36.     if (cardsToPlay.Count < 5) {
37.         return false;
38.     }
```

```
39.     return true;
40. }
```

Kode Sumber 4.12 Greedy untuk *CardPacket Full House*

4.4.7. Four of a Kind

```
1.  public bool findFourOfAKind() {
2.      for (int x = 0; x < handCards.Count - 3; x++) {
3.          if (handCards[x] / 4 == handCards[x + 1] / 4 && handCards[x] /
    4 == handCards[x + 2] / 4 && handCards[x] / 4 == handCards[x + 3] / 4)
    {
4.              cardsToPlay.Add(handCards[x]);
5.              cardsToPlay.Add(handCards[x + 1]);
6.              cardsToPlay.Add(handCards[x + 2]);
7.              cardsToPlay.Add(handCards[x + 3]);
8.              if (Handler.cardPacket == "FourOfAKind") {
9.                  if (HandlerObject.GetComponent < Handler > ().fieldCard
    s[0] / 4 == HandlerObject.GetComponent < Handler > ().fieldCards[1] / 4
    ) {
10.                     if (cardsToPlay[0] > HandlerObject.GetComponent < H
    andler > ().fieldCards[0]) {
11.                         break;
12.                     } else {
13.                         cardsToPlay.Clear();
14.                     }
15.                 } else {
16.                     if (cardsToPlay[0] > HandlerObject.GetComponent < H
    andler > ().fieldCards[4]) {
17.                         break;
18.                     } else {
19.                         cardsToPlay.Clear();
20.                     }
21.                 }
22.             } else {
23.                 break;
24.             }
25.         }
26.     }
27.     if (cardsToPlay.Count < 4) {
28.         return false;
29.     }
30.     for (int x = 0; x < handCards.Count; x++) {
31.         if (handCards[x] / 4 != cardsToPlay[0] / 4) {
```

```
            cardsToPlay.Add(handCards[x]);
            break;
        }
    }
    if (cardsToPlay.Count < 5) {
        return false;
    }
    return true;
}
```

Kode Sumber 4.13 Greedy untuk *CardPacket Four of a Kind*

4.4.8. Straight Flush

```
public bool findStraightFlush() {
    for (int x = 0; x < handCards.Count - 4; x++) {
        cardsToPlay.Add(handCards[x]);
        for (int y = x + 1; y < handCards.Count; y++) {
            if (handCards[y] / 4 == Mathf.Max(cardsToPlay.ToArray()) / 4 + 1) {
                cardsToPlay.Add(handCards[y]);
            }
            if (cardsToPlay.Count == 5 || handCards[y] / 4 > Mathf.Max(cardsToPlay.ToArray()) / 4 + 1) {
                break;
            }
        }
        if (cardsToPlay.Count == 5) {
            bool temp = true;
            for (int z = 0; z < cardsToPlay.Count - 1; z++) {
                if (cardsToPlay[z] % 4 != cardsToPlay[z + 1] % 4) {
                    temp = false;
                    break;
                }
            }
            if (temp) {
                if (cardsToPlay[0] % 4 > HandlerObject.GetComponent < Handler > ().fieldCards[0] % 4) {
                    return true;
                } else if (cardsToPlay[0] % 4 == HandlerObject.GetComponent < Handler > ().fieldCards[0] % 4) {
                    if (Mathf.Max(cardsToPlay.ToArray()) > Mathf.Max(HandlerObject.GetComponent < Handler > ().fieldCards.ToArray())) {
                        return true;
```

```
                    } else {
                        cardsToPlay.Clear();
                    }
                } else {
                    cardsToPlay.Clear();
                }
            } else {
                cardsToPlay.Clear();
            }
        } else {
            cardsToPlay.Clear();
        }
    }
    return false;
}
```

Kode Sumber 4.14 Greedy untuk *CardPacket Straight Flush*

BAB V
PENGUJIAN DAN EVALUASI

5.1. Lingkungan Pengujian

Lingkungan pelaksanaan uji coba meliputi perangkat keras dan perangkat lunak yang akan digunakan pada sistem ini. Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam rangka uji coba perangkat lunak ini dicantumkan pada Tabel 5.1.

Tabel 5.1 Lingkungan pengujian

Perangkat Keras	Meizu MX6 CPU: Deca-core (2x2.3GHz Cortex-A72, 4x1.9GHz Cortex-A53, 4x1.4GHz Cortex-A53) Memori : 4.00 GB
Perangkat Lunak	Sistem Operasi Android 6.0

5.2. Pengujian Fitur

Untuk mengetahui kesesuaian keluaran dari tiap tahap dan langkah penggunaan fitur terhadap skenario yang dipersiapkan, maka dibutuhkan pengujian fitur.

Pengujian ini dilakukan untuk menguji apakah fitur yang diidentifikasi benar-benar diimplementasikan dan bekerja sebagaimana seharusnya. Pengujian juga dilakukan untuk mengetahui kesesuaian setiap tahapan atau langkah penggunaan fitur terhadap skenario yang dipersiapkan.

5.2.1. Skenario Pengujian

Skenario pengujian fitur digunakan untuk memberikan tahap-tahap dalam pengujian sistem. Skenario tertera pada Tabel 5.2.

Tabel 5.2 Pengujian aplikasi permainan

Kondisi Awal	Pengguna berada pada layar Main Menu
Prosedur Pengujian	Pengguna memilih mode permainan dan menyelesaikan permainan sesuai dengan mode yang dipilih
Hasil yang diharapkan	Pengguna berhasil menyelesaikan permainan dan fitur permainan berjalan dengan lancar.
Hasil yang diperoleh	Pengguna berhasil menyelesaikan permainan dan fitur berjalan lancar.
Kesimpulan	Pengujian berhasil

5.2.1.1. Pengujian Main Menu dan Difficulty Selection

Pengujian dimulai ketika pengguna telah masuk ke layar *Main Menu* seperti yang terlihat pada Gambar 5.1. Pemain menekan tombol *Single Player* untuk memunculkan tampilan pemilihan tangkat kesulitan.

Gambar 5.1 Tampilan *Main Menu*

Setelah menekan tombol *Single Player* maka sistem menampilkan tampilan pemilihan tangkat kesulitan seperti yang terlihat pada Gambar 5.2.

Gambar 5.2 Tampilan *Difficulty Selection*

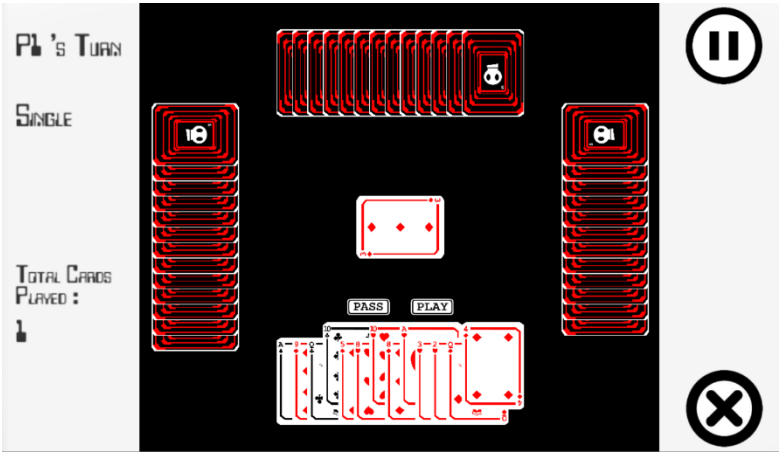
Setelah masuk ke muncul tampilan *difficulty selection*, pemain dapat memilih tingkat kesulitan *Computer Player* yang ingin dilawan dengan memilih salah satu opsi yang ada. Saat pemain memilih tingkat kesulitan yang diinginkan maka akan muncul tampilan gameplay dengan *Computer Player* yang memiliki tingkat kesulitan yang dipilih.

Setelah melakukan pengujian, sistem berhasil masuk ke scene permainan dan memunculkan *Computer Player* sesuai dengan tingkat kesulitan yang dipilih. Sehingga dapat disimpulkan bahwa pengujian untuk layar *Main Menu* dan *Difficulty Selection* berhasil.

5.2.1.2. Pengujian Aksi Pemilihan Kartu

Pengujian dimulai ketika pemain telah masuk ke *gameplay scene*. Pemain dapat memilih kartu yang diinginkan dengan cara menekannya. Setelah ditekan maka kartu yang dipilih akan terangkat sedikit ke atas untuk menunjukkan bahwa kartu itu telah dipilih. Setelah pengujian, kartu yang dipilih

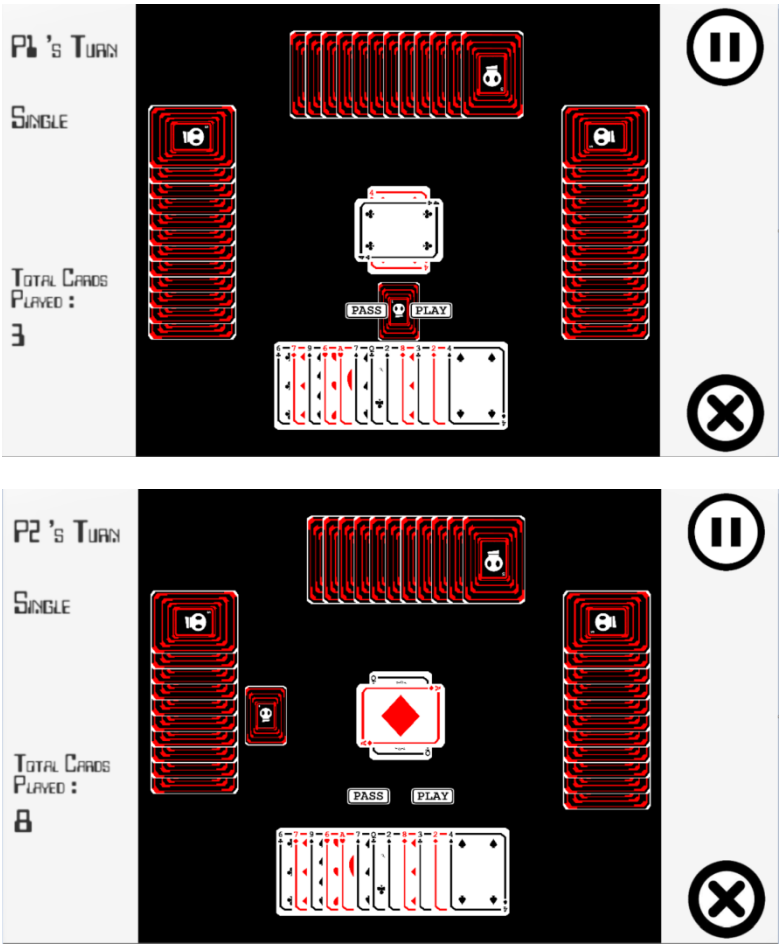
berhasil terangkat sesuai, sehingga dapat disimpulkan bahwa pengujian untuk aksi pemilihan kartu oleh pemain berhasil. Berikut hasil pengujian ini dapat dilihat pada Gambar 5.3.



Gambar 5.3 Tampilan pemilihan kartu

5.2.1.3. Pengujian *Play* dan *Pass*

Pengujian dimulai ketika pemain telah masuk ke *gameplay scene*. Pemain dapat memainkan kartu yang telah dipilih atau bisa juga melewatkan giliran mereka. Setelah diuji disimpulkan bahwa *Play* dan *Pause* berhasil. Berikut hasil dari pengujian ini dapat dilihat pada Gambar 5.4.



Gambar 5.4 Tampilan saat pemain memilih *play* (atas) dan *pass* (bawah)

5.2.1.4. Pengujian Menang dan Kalah

Pengujian ini dimulai dengan menyelesaikan permainan. Saat pemain telah menghabiskan kartu di tangan dan memenangkan permainan maka *window* atau tampilan menang

akan muncul, menandakan pemain telah sukses memenangkan permainan.Tampilan pemain menang dan tampilan stage menang muncul dapat dilihat pada Gambar 5.8.

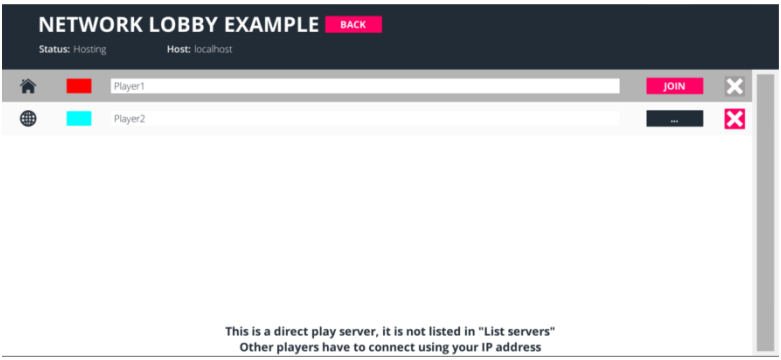
Gambar 5.5 Tampilan saat pemain menang

Pemain akan dinyatakan kalah saat *Computer Player* terlebih dahulu menghabiskan kartu di tangan mereka. Saat kartu di tangan salah satu *Computer Player* habismaka akan muncul *window* atau tampilan kalah. Tampilan pemain kalah dapat dilihat padaGambar 5.6.

Gambar 5.6 Pemain (P1) kalah dan *Computer Player* (P3) memenangkan permainan

5.2.1.5. Pengujian Multiplayer Lobby Connection

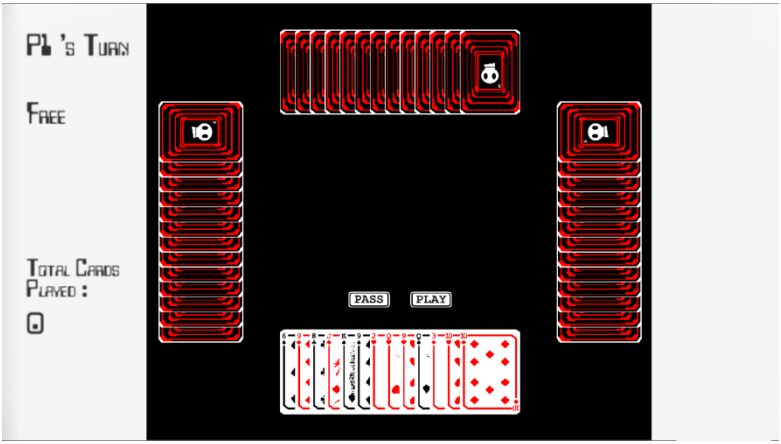
Pengujian ini dimulai dengan pemilihan opsi *Multiplayer* pada main menu. Terdapat 3 pilihan yang dapat dilakukan pemain, antara lain *host*, *dedicated server*, dan *join*. Untuk *host* artinya perangkat pemain berperan sebagai *server* dan juga sebagai *client*. Untuk lebih jelas dapat di lihat di gambar 5.7.



Gambar 5.7 Multiplayer Lobby Connection

5.2.1.6. Pengujian Multiplayer Gameplay

Pengujian ini dimulai dengan ketika seluruh pemain lain di dalam *room* kecuali *room master* sudah dalam status *ready*. Jika sudah maka *room master* akan memulai permainan dengan memilih pilihan *join*. Setelah itu maka setiap player akan memulai permainan.



Gambar 5.8 Multiplayer Gameplay

5.2.2. Hasil Pengujian

Hasil uji fitur yang telah dilakukan berdasarkan skenario sebelumnya, menunjukkan bahwa semua fitur permainan berjalan dengan baik dan sesuai dengan sebagaimana mestinya yang telah dibuat pada tahap perancangan. Hasil uji fitur dapat dilihat pada Tabel 5.3.

Tabel 5.3 Hasil pengujian fitur

No	Pengujian	Hasil Pengujian
1	*Main Menu* dan *Difficulty Selection*	Berhasil

2	Aksi pemilihan kartu	Berhasil
3	*Play* dan *pass*	Berhasil
4	Menang dan Kalah	Berhasil
5	*Multiplayer Lobby Connection*	Berhasil
6	*Multiplayer Gameplay*	Berhasil

5.3. Pengujian Penerapan Algoritma Greedy

Pengujian penerapan algoritma Greedy pada game Capsa Banting mencakup seberapa efektif algoritma tersebut jika di terapkan pada *Computer Player* pada tingkat kesulitan yang berbeda-beda.

5.3.1. Skenario Pengujian

Pengujian dilakukan pada 30 kali permainan ‘Capsa Banting’ dengan pemainnya merupakan 4 *Computer Player* yang menerapkan algoritma Greedy. Setiap *Computer Player* memiliki tingkat kesulitan yang berbeda. Dikarenakan hanya terdapat 3 tingkat kesulitan sedangkan terdapat 4 *Computer Player* maka satu *Computer Player* akan berganti tingkat kesulitan setiap 10 pengujian.

5.3.2. Hasil Pengujian

Hasil yang diuji merupakan jumlah kemenangan dari tiap *Computer Player*. Apakah *Computer Player* dengan tingkat kesulitan lebih tinggi akan lebih sering memenangkan permainan.

Pengujian dilakukan pada seluruh *Computer Player* yang menerapkan algoritma Greedysebanyak 30 kali permainan. Hasil pengujian 30 permainan dapat dilihat pada tabel 5.4.

Tabel 5.4 Hasil pengujian

No	P1	P2	P3	P4	Pemenang
1	Easy	Easy	Medium	Hard	P3
2	Easy	Easy	Medium	Hard	P3
3	Easy	Easy	Medium	Hard	P4

4	Easy	Easy	Medium	Hard	P1
5	Easy	Easy	Medium	Hard	P4
6	Easy	Easy	Medium	Hard	P4
7	Easy	Easy	Medium	Hard	P4
8	Easy	Easy	Medium	Hard	P1
9	Easy	Easy	Medium	Hard	P4
10	Easy	Easy	Medium	Hard	P3
11	Medium	Easy	Medium	Hard	P4
12	Medium	Easy	Medium	Hard	P2
13	Medium	Easy	Medium	Hard	P4
14	Medium	Easy	Medium	Hard	P3
15	Medium	Easy	Medium	Hard	P2
16	Medium	Easy	Medium	Hard	P4
17	Medium	Easy	Medium	Hard	P4
18	Medium	Easy	Medium	Hard	P2
19	Medium	Easy	Medium	Hard	P3
20	Medium	Easy	Medium	Hard	P1
21	Hard	Easy	Medium	Hard	P3
22	Hard	Easy	Medium	Hard	P3
23	Hard	Easy	Medium	Hard	P3
24	Hard	Easy	Medium	Hard	P1
25	Hard	Easy	Medium	Hard	P4
26	Hard	Easy	Medium	Hard	P1
27	Hard	Easy	Medium	Hard	P1
28	Hard	Easy	Medium	Hard	P1
29	Hard	Easy	Medium	Hard	P4
30	Hard	Easy	Medium	Hard	P1

Hasilnya adalah *Computer Player* dengan tingkat kesulitan *hard* memenangkan 16 permainan, *medium* 9 permainan, *easy* 5 permainan. Dari hasil uji ini dapat disimpulkan semakin tinggi tingkat kesulitan *Computer Player* maka semakin baik pula penggunaan algoritma Greedy dalam pemilihan kartunya. Pada hasil tersebut dapat dilihat bahwa *Computer Player* dengan tingkat kesulitan *hard* tidak selalu memenangkan pertandingan. Hal ini dikarenakan karena pada saat pembagian kartu *Computer Player* dengan tingkat kesulitan *hard* mendapat kartu yang kurang bagus dibandingkan dengan *Computer Player*

dengan tingkat kesulitan lain. Salah satu contoh kasus pembagian kartu dimana *Computer Easy* memenangkan permainan adalah :

Tabel 5.5 Contoh Kasus Pembagian Kartu

No	P1 (Easy)	P2 (Easy)	P3 (Medium)	P4 (Hard)
1	3 Diamond	4 Diamond	3 Hearts	3 Club
2	6 Spade	4 Club	3 Spade	5 Club
3	10 Diamond	4 Spade	4 Hearts	5 Hearts
4	10 Hearts	5 Diamond	6 Club	5 Spade
5	Jack Diamond	6 Diamond	8 Club	7 Club
6	Jack Club	6 Hearts	8 hearts	7 Spade
7	Queen Diamond	7 Diamond	9 Club	9 Hearts
8	King Hearts	7 Hearts	9 Spade	10 Club
9	King Spade	8 Diamond	10 Spade	Jack Hearts
10	Ace Diamond	8 Spade	Queen Club	Jack Spade
11	Ace Hearts	9 Diamond	King Club	Queen Hearts
12	2 Hearts	Queen Spade	Ace Club	King Diamond
13	2 Spade	2 Club	Ace Spade	2 Diamond
Total Nilai	**463**	**252**	**333**	**330**

Jika nilai seluruh kartu di jumlah dengan aturan 3 Diamond = 1 dan 2 Spade = 52, maka dapat dilihat bahwa total nilai kartu P1 adalah yang tertinggi di antara Computer Player lain. Selain itu pada Computer Player hard hanya terdapat 1 kombinasi 5 kartu, serta 2 kombinasi 2 kartu. Yang dapat mengeluarkan kombinasi 5 kartu hanyalah Computer Player dengan tingkat kesulitan hard oleh karena itu Computer Player hard harus dapat memenangkan 1 putaran tanpa merusak kombinasi 5 kartu yang ada di tangannya, dimana merupakan hal yang sulit kartu pada pemain lain memiliki nilai yang lebih besar dibanding kan kartu *Computer Player* hard.

5.4. Pengujian Pengguna

Pengujian pengguna adalah untuk mendapatkan informasi mengenai pendapat pengguna mengenai game ‘Capsa Banting’. Pengujian dilakukan pada tiga pengguna.

5.4.1. Skenario Pengujian

Pengujian dilakukan oleh pengguna (pemain) dengan memainkan game ‘Capsa Banting’. Pemain memainkan masing-masing tingkat kesulitan berulang-ulang. Kemudian pemain diberikan *form* berisi beberapa pertanyaan. Pertanyaan yang diberikan meliputi pendapatnya mengenai game ‘Capsa Banting’, pendapat pengguna mengenai tingkat kesulitan yang diterapkan, dan masukan untuk pengembangan aplikasi permainan lebih lanjut.

5.4.2. Hasil Pengujian

Setelah dilakukan pengujian oleh pengguna, didapatkan beberapa informasi. Informasi tingkat kesulitan yang berulang kali dicoba pemain ditunjukkan pada Tabel 5.5.

Tabel 5.6 Hasil Pengujian Pengguna

Pengguna	Tingkat Kesulitan yang Dicoba	Jumlah Dimainkan	Jumlah Menang
1	Easy	3	3
	Medium	3	3
	Hard	3	1
2	Easy	3	3
	Medium	3	1
	Hard	3	1
3	Easy	3	3
	Medium	3	2
	Hard	3	0

Adanya tingkat kesulitan pada permainan ‘Capsa Banting’ membuat permainan ini menjadi lebih menantang. Ketiga pengguna juga berpendapat bahwa tingkat kesulitan ini membantu untuk mengenal permainan secara perlahan.

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut.

6.1. Kesimpulan

Dari proses penerapan algoritma Greedy untuk pemilihan kartuyang akan di keluarkan *Computer Player* pada permainan 'Capsa Banting' dapat diambil kesimpulan sebagai berikut:

1. 'Capsa Banting' berhasil dirancang dengan baik menerapkan rancangan *gameplay* permainan yang ditetapkan dan fitur dalam permainan dapat berjalan dengan sebagaimana mestinya.
2. Aturan pemilihan kartu menggunakan algoritma Greedy berhasil didapatkan dengan menyesuaikan algoritma dan *gameplay* game 'Capsa Banting'.
3. Tingkat kesulitan sesuai dengan yang diharapkan dimana semakin tinggi tingkat kesulitan maka semakin sulit bagi pemain untuk memenangkan permainan.
4. Penerapan algoritma Greedy dalam pemilihan kartu dalam game diterapkan dalam bentuk script pada game engine, dan permainan dapat berjalan sebagaimana mestinya dalam perangkat mobile.

6.2. Saran

Saran yang diberikan terkait pengembangan pada Tugas Akhir ini berdasarkan perancangan, implementasi, dan uji coba adalah:

1. Meningkatkan kualitas visual permainan agar lebih menarik bagi pemain.

2. Memperbanyak jumlah pengujian terhadap setiap tingkat kesulitan.
3. Untuk pengembangan lebih lanjut disarankan agar menggabungkan penggunaan algoritma Greedy dengan algoritma lain untuk mengingkatkan kesulitan dari *Computer Player*, serta memberi *Computer Player* optimasi keputusan bukan hanya untuk kartu apa yang dikeluarkan tapi untuk kapan harus mengeluarkan kartu.

DAFTAR PUSTAKA

[1] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2001). *Introduction to Algorithms*. London: The MIT Press.

[2] Dix, A., Finlay, J., Abowd, G. D., & Beale, R. (2004). *Human-Computer Interaction (Third Edition).* Harlow: Pearson Education Limited.

[3] Unity Technologies. (2017, September 29). *CREATE THE GAMES YOU LOVE WITH UNITY*. Retrieved from Unity - Game Engine: http://unity3d.com/unity [Diakses 16April 2018].

[4] Hejlsberg, A. (2008). *The A-Z of Programming Languages* . October: 1.

[5] Sim, A. X. (2013). *Algoritma Greedy - Dasar Analisis Algoritma*. Retrieved from bertzzie's rant - Home: https://bertzzie.com/knowledge/analisis-algoritma/Greedy.html[Diakses 16April 2018].

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Penulis lahir di Medan, 5April 1997, merupakan anak kedua dari dua bersaudara. Dalam menjalani pendidikan semasa hidup, penulis menempuh pendidikan di TK Santo Yoseph Medan, SD Santo Yoseph Medan, SMP Santo Thomas 1 Medan, SMA Sutomo 1 Medan dan S1 Departemen Informatika Institut Teknologi Sepuluh Nopember (ITS) pada rumpun Interaksi Grafika dan Seni (IGS).

Selama menjadi mahasiswa, penulis aktif dalam beberapa organisasi kemahasiswaan antara lain Staff Minat Bakat Himpunan Mahasiswa Teknik Komputer Informatika ITS (HMTC), dan 2 tahun berturut-turut menjadi Staff National Programming Contest (NPC) Schematics ITS.

[Halaman ini sengaja dikosongkan]

LAMPIRAN A

```
1.  public class P1AI: MonoBehaviour {
2.      public List < int > handCards;
3.      public List < int > cardsToPlay;
4.      public GameObject spinUp;
5.      public GameObject cardPlayed;
6.      public GameObject HandlerObject;
7.      public GameObject CheckerObject;
8.      string msg;
9.      public bool pass = false;
10.     public bool played = false;
11.     public AudioSource cardSpin;
12.     public AudioSource lose; // Use this for initialization
13.     void Start() {
14.             pass = false;
15.             played = false;
16.         } // Update is called once per frame
17.     void Update() {
18.         handCards.Sort();
19.         if (Handler.turn == 0 && played == false) {
20.             if (!pass) {
21.                 if (Handler.cardPacket == "Free") {
22.                     if (Handler.gameStart) {
23.                         if (GameObject.Find("DifficultyScript").GetComp
    onent < Difficulty > ().difficulty == "Easy") {
24.                             cardsToPlay.Add(handCards[Random.Range(0, h
    andCards.Count)]);
25.                             toField();
26.                             Handler.cardPacket = "Single";
27.                         } else if (GameObject.Find("DifficultyScript").
    GetComponent < Difficulty > ().difficulty == "Medium") {
28.                             cardsToPlay.Add(handCards[0]);
29.                             toField();
30.                             Handler.cardPacket = "Single";
31.                         } else if (GameObject.Find("DifficultyScript").
    GetComponent < Difficulty > ().difficulty == "Hard") {
32.                             if (findStraight()) {
33.                                 toField();
34.                                 Handler.cardPacket = "Straight";
35.                             } else if (findFlush()) {
36.                                 toField();
```

```
37.                                 Handler.cardPacket = "Flush";
38.                             } else if (findFullHouse()) {
39.                                 toField();
40.                                 Handler.cardPacket = "FullHouse";
41.                             } else if (findFourOfAKind()) {
42.                                 toField();
43.                                 Handler.cardPacket = "FourOfAKind";
44.                             } else if (findStraightFlush()) {
45.                                 toField();
46.                                 Handler.cardPacket = "StraightFlush";
47.                             } else if (findRoyalStraightFlush()) {
48.                                 toField();
49.                                 Handler.cardPacket = "RoyalStraightFlus
    h";
50.                             } else if (findThrees()) {
51.                                 toField();
52.                                 Handler.cardPacket = "Threes";
53.                             } else if (findPair()) {
54.                                 toField();
55.                                 Handler.cardPacket = "Pair";
56.                             } else {
57.                                 cardsToPlay.Add(handCards[0]);
58.                                 toField();
59.                                 Handler.cardPacket = "Single";
60.                             }
61.                         }
62.                     } else {
63.                         Handler.gameStart = true;
64.                         cardsToPlay.Add(0);
65.                         toField();
66.                         Handler.cardPacket = "Single";
67.                     }
68.                 } else if (Handler.cardPacket == "Single") {
69.                     bool play = false;
70.                     for (int x = 0; x < handCards.Count; x++) {
71.                         if (handCards[x] > HandlerObject.GetComponent <
    Handler > ().fieldCards[0]) {
72.                             play = true;
73.                             cardsToPlay.Add(handCards[x]);
74.                             toField();
75.                             break;
76.                         }
77.                     }
```

```
78.                     if (!play) {
79.                         passTurn();
80.                     }
81.                 } else if (Handler.cardPacket == "Pair" && handCards.Co
    unt >= 2) {
82.                     if (findPair()) {
83.                         toField();
84.                     } else {
85.                         passTurn();
86.                     }
87.                 } else if (Handler.cardPacket == "Threes" && handCards.
    Count >= 3) {
88.                     if (findThrees()) {
89.                         toField();
90.                     } else {
91.                         passTurn();
92.                     }
93.                 } else if (Handler.cardPacket == "Straight" && handCard
    s.Count >= 5) {
94.                     if (findStraight()) {
95.                         toField();
96.                     } else if (findFlush()) {
97.                         toField();
98.                         Handler.cardPacket = "Flush";
99.                     } else if (findFullHouse()) {
100.                            toField();
101.                            Handler.cardPacket = "FullHouse";
102.                        } else if (findFourOfAKind()) {
103.                            toField();
104.                            Handler.cardPacket = "FourOfAKind";
105.                        } else if (findStraightFlush()) {
106.                            toField();
107.                            Handler.cardPacket = "StraightFlush";
108.                        } else if (findRoyalStraightFlush()) {
109.                            toField();
110.                            Handler.cardPacket = "RoyalStraightFlush";

111.                        } else {
112.                            passTurn();
113.                        }
114.                    } else if (Handler.cardPacket == "Flush") {
115.                        if (findFlush()) {
116.                            toField();
```

```
117.                        } else if (findFullHouse()) {
118.                            toField();
119.                            Handler.cardPacket = "FullHouse";
120.                        } else if (findFourOfAKind()) {
121.                            toField();
122.                            Handler.cardPacket = "FourOfAKind";
123.                        } else if (findStraightFlush()) {
124.                            toField();
125.                            Handler.cardPacket = "StraightFlush";
126.                        } else if (findRoyalStraightFlush()) {
127.                            toField();
128.                            Handler.cardPacket = "RoyalStraightFlush";

129.                        } else {
130.                            passTurn();
131.                        }
132.                    } else if (Handler.cardPacket == "FullHouse") {
133.                        if (findFullHouse()) {
134.                            toField();
135.                        } else if (findFourOfAKind()) {
136.                            toField();
137.                            Handler.cardPacket = "FourOfAKind";
138.                        } else if (findStraightFlush()) {
139.                            toField();
140.                            Handler.cardPacket = "StraightFlush";
141.                        } else if (findRoyalStraightFlush()) {
142.                            toField();
143.                            Handler.cardPacket = "RoyalStraightFlush";

144.                        } else {
145.                            passTurn();
146.                        }
147.                    } else if (Handler.cardPacket == "FourOfAKind") {

148.                        if (findFourOfAKind()) {
149.                            toField();
150.                        } else if (findStraightFlush()) {
151.                            toField();
152.                            Handler.cardPacket = "StraightFlush";
153.                        } else if (findRoyalStraightFlush()) {
154.                            toField();
155.                            Handler.cardPacket = "RoyalStraightFlush";
```

```
156.                         } else {
157.                             passTurn();
158.                         }
159.                     } else if (Handler.cardPacket == "StraightFlush")
    {
160.                         if (findStraightFlush()) {
161.                             toField();
162.                         } else if (findRoyalStraightFlush()) {
163.                             toField();
164.                             Handler.cardPacket = "RoyalStraightFlush";

165.                         } else {
166.                             passTurn();
167.                         }
168.                     } else if (Handler.cardPacket == "RoyalStraightFlu
    sh") {
169.                         passTurn();
170.                     } else {
171.                         passTurn();
172.                     }
173.                 } else {
174.                     Debug.Log("P1 Already Passed");
175.                     Handler.turn++;
176.                 }
177.             }
178.         }
179.         public bool findPair() {
180.             for (int x = 1; x < handCards.Count; x++) {
181.                 if (handCards[x -
    1] / 4 == handCards[x] / 4 && handCards[x] > Mathf.Max(HandlerObject.G
    etComponent < Handler > ().fieldCards.ToArray())) {
182.                     cardsToPlay.Add(handCards[x - 1]);
183.                     cardsToPlay.Add(handCards[x]);
184.                     return true;
185.                 }
186.             }
187.             return false;
188.         }
189.         public bool findThrees() {
190.             for (int x = 2; x < handCards.Count; x++) {
191.                 if (handCards[x -
    2] / 4 == handCards[x] / 4 && handCards[x -
```

```
    1] / 4 == handCards[x] / 4 && handCards[x] > Mathf.Max(HandlerObject.G
    etComponent < Handler > ().fieldCards.ToArray())) {
192.                        cardsToPlay.Add(handCards[x - 2]);
193.                        cardsToPlay.Add(handCards[x - 1]);
194.                        cardsToPlay.Add(handCards[x]);
195.                        return true;
196.                    }
197.                }
198.                return false;
199.            }
200.            public bool findStraight() {
201.                if (Handler.cardPacket != "Straight") {
202.                    return false;
203.                }
204.                for (int x = 0; x < handCards.Count - 4; x++) {
205.                    cardsToPlay.Add(handCards[x]);
206.                    for (int y = x + 1; y < handCards.Count; y++) {
207.                        if (handCards[y] / 4 == Mathf.Max(cardsToPlay.ToAr
    ray()) / 4 + 1) {
208.                            cardsToPlay.Add(handCards[y]);
209.                        }
210.                        if (cardsToPlay.Count == 5 || handCards[y] / 4 > M
    athf.Max(cardsToPlay.ToArray()) / 4 + 1) {
211.                            break;
212.                        }
213.                    }
214.                    if (cardsToPlay.Count == 5 && Mathf.Max(cardsToPlay.To
    Array()) > Mathf.Max(HandlerObject.GetComponent < Handler > ().fieldCar
    ds.ToArray())) {
215.                        bool temp = false;
216.                        for (int z = 0; z < cardsToPlay.Count -
     1; z++) {
217.                            if (cardsToPlay[z] % 4 != cardsToPlay[z + 1] %
     4) {
218.                                temp = true;
219.                                break;
220.                            }
221.                        }
222.                        if (temp) {
223.                            return true;
224.                        } else {
225.                            cardsToPlay.Clear();
226.                        }
```

```
227.                 } else {
228.                     cardsToPlay.Clear();
229.                 }
230.             }
231.             return false;
232.         }
233.         public bool findFlush() {
234.             for (int x = 0; x < handCards.Count - 4; x++) {
235.                 cardsToPlay.Add(handCards[x]);
236.                 for (int y = x + 1; y < handCards.Count; y++) {
237.                     if (handCards[y] % 4 == cardsToPlay[0] % 4) {
238.                         cardsToPlay.Add(handCards[y]);
239.                     }
240.                     if (cardsToPlay.Count == 5) {
241.                         break;
242.                     }
243.                 }
244.                 if (cardsToPlay.Count == 5) {
245.                     if (Handler.cardPacket != "Flush" || (Handler.card
    Packet == "Flush" && Mathf.Max(cardsToPlay.ToArray()) > Mathf.Max(Handl
    erObject.GetComponent < Handler > ().fieldCards.ToArray()))) {
246.                         bool temp = false;
247.                         for (int z = 0; z < cardsToPlay.Count -
     1; z++) {
248.                             if ((cardsToPlay[z] / 4) + 1 != cardsToPla
    y[z + 1] / 4) {
249.                                 temp = true;
250.                                 break;
251.                             }
252.                         }
253.                         if (temp) {
254.                             return true;
255.                         } else {
256.                             cardsToPlay.Clear();
257.                         }
258.                     } else {
259.                         cardsToPlay.Clear();
260.                     }
261.                 } else {
262.                     cardsToPlay.Clear();
263.                 }
264.             }
265.             return false;
```

```
266.         }
267.         public bool findFullHouse() {
268.             for (int x = 0; x < handCards.Count - 2; x++) {
269.                 if (handCards[x] / 4 == handCards[x + 1] / 4 && handCa
    rds[x] / 4 == handCards[x + 2] / 4) {
270.                     cardsToPlay.Add(handCards[x]);
271.                     cardsToPlay.Add(handCards[x + 1]);
272.                     cardsToPlay.Add(handCards[x + 2]);
273.                     if (Handler.cardPacket == "FullHouse") {
274.                         if (HandlerObject.GetComponent < Handler > ().
    fieldCards[1] / 4 == HandlerObject.GetComponent < Handler > ().fieldCar
    ds[2] / 4) {
275.                             if (cardsToPlay[0] > HandlerObject.GetComp
    onent < Handler > ().fieldCards[0]) {
276.                                 break;
277.                             } else {
278.                                 cardsToPlay.Clear();
279.                             }
280.                         } else {
281.                             if (cardsToPlay[0] > HandlerObject.GetComp
    onent < Handler > ().fieldCards[4]) {
282.                                 break;
283.                             } else {
284.                                 cardsToPlay.Clear();
285.                             }
286.                         }
287.                     } else {
288.                         break;
289.                     }
290.                 }
291.             }
292.             if (cardsToPlay.Count < 3) {
293.                 return false;
294.             }
295.             for (int x = 0; x < handCards.Count - 1; x++) {
296.                 if (handCards[x] / 4 == handCards[x + 1] / 4 && handCa
    rds[x] / 4 != cardsToPlay[0] / 4) {
297.                     cardsToPlay.Add(handCards[x]);
298.                     cardsToPlay.Add(handCards[x + 1]);
299.                     break;
300.                 }
301.             }
302.             if (cardsToPlay.Count < 5) {
```

```
303.                 return false;
304.             }
305.             return true;
306.         }
307.         public bool findFourOfAKind() {
308.             for (int x = 0; x < handCards.Count - 3; x++) {
309.                 if (handCards[x] / 4 == handCards[x + 1] / 4 && handCa
    rds[x] / 4 == handCards[x + 2] / 4 && handCards[x] / 4 == handCards[x +
     3] / 4) {
310.                     cardsToPlay.Add(handCards[x]);
311.                     cardsToPlay.Add(handCards[x + 1]);
312.                     cardsToPlay.Add(handCards[x + 2]);
313.                     cardsToPlay.Add(handCards[x + 3]);
314.                     if (Handler.cardPacket == "FourOfAKind") {
315.                         if (HandlerObject.GetComponent < Handler > ().
    fieldCards[0] / 4 == HandlerObject.GetComponent < Handler > ().fieldCar
    ds[1] / 4) {
316.                             if (cardsToPlay[0] > HandlerObject.GetComp
    onent < Handler > ().fieldCards[0]) {
317.                                 break;
318.                             } else {
319.                                 cardsToPlay.Clear();
320.                             }
321.                         } else {
322.                             if (cardsToPlay[0] > HandlerObject.GetComp
    onent < Handler > ().fieldCards[4]) {
323.                                 break;
324.                             } else {
325.                                 cardsToPlay.Clear();
326.                             }
327.                         }
328.                     } else {
329.                         break;
330.                     }
331.                 }
332.             }
333.             if (cardsToPlay.Count < 4) {
334.                 return false;
335.             }
336.             for (int x = 0; x < handCards.Count; x++) {
337.                 if (handCards[x] / 4 != cardsToPlay[0] / 4) {
338.                     cardsToPlay.Add(handCards[x]);
339.                     break;
```

```
340.                 }
341.             }
342.             if (cardsToPlay.Count < 5) {
343.                 return false;
344.             }
345.             return true;
346.         }
347.         public bool findStraightFlush() {
348.             for (int x = 0; x < handCards.Count - 4; x++) {
349.                 cardsToPlay.Add(handCards[x]);
350.                 for (int y = x + 1; y < handCards.Count; y++) {
351.                     if (handCards[y] / 4 == Mathf.Max(cardsToPlay.ToAr
    ray()) / 4 + 1) {
352.                         cardsToPlay.Add(handCards[y]);
353.                     }
354.                     if (cardsToPlay.Count == 5 || handCards[y] / 4 > M
    athf.Max(cardsToPlay.ToArray()) / 4 + 1) {
355.                         break;
356.                     }
357.                 }
358.                 if (cardsToPlay.Count == 5) {
359.                     bool temp = true;
360.                     for (int z = 0; z < cardsToPlay.Count -
     1; z++) {
361.                         if (cardsToPlay[z] % 4 != cardsToPlay[z + 1] %
     4) {
362.                             temp = false;
363.                             break;
364.                         }
365.                     }
366.                     if (temp) {
367.                         if (cardsToPlay[0] % 4 > HandlerObject.GetComp
    onent < Handler > ().fieldCards[0] % 4) {
368.                             return true;
369.                         } else if (cardsToPlay[0] % 4 == HandlerObject
    .GetComponent < Handler > ().fieldCards[0] % 4) {
370.                             if (Mathf.Max(cardsToPlay.ToArray()) > Mat
    hf.Max(HandlerObject.GetComponent < Handler > ().fieldCards.ToArray()))
     {
371.                                 return true;
372.                             } else {
373.                                 cardsToPlay.Clear();
374.                             }
```

```
375.                         } else {
376.                             cardsToPlay.Clear();
377.                         }
378.                     } else {
379.                         cardsToPlay.Clear();
380.                     }
381.                 } else {
382.                     cardsToPlay.Clear();
383.                 }
384.             }
385.             return false;
386.         }
387.         public bool findRoyalStraightFlush() {
388.             if (handCards.Contains(47) && handCards.Contains(43) && ha
    ndCards.Contains(39) && handCards.Contains(35) && handCards.Contains(31
    )) {
389.                 cardsToPlay.Add(47);
390.                 cardsToPlay.Add(43);
391.                 cardsToPlay.Add(39);
392.                 cardsToPlay.Add(35);
393.                 cardsToPlay.Add(31);
394.                 return true;
395.             } else {
396.                 return false;
397.             }
398.         }
399.         public void passTurn() {
400.             Debug.Log("P1 Passed");
401.             pass = true;
402.             HandlerObject.GetComponent < Handler > ().playerPassed.Add
    ("P1");
403.             if (HandlerObject.GetComponent < Handler > ().playerPassed
    .Count >= 3) {
404.                 HandlerObject.GetComponent < Handler > ().NewRound();

405.             } else {
406.                 Handler.turn++;
407.             }
408.         }
409.         public void toField() {
410.             cardSpin.Play();
411.             HandlerObject.GetComponent < Handler > ().fieldCards.Clear
    ();
```

```
412.                cardsToPlay.Sort();
413.                for (int x = 0; x < cardsToPlay.Count; x++) {
414.                    Handler.cardsOnField++;
415.                    handCards.Remove(cardsToPlay[x]);
416.                }
417.                for (int i = 0; i < cardsToPlay.Count; i++) {
418.                    HandlerObject.GetComponent < Handler > ().fieldCards.A
    dd(cardsToPlay[i]);
419.                    msg += ", " + cardsToPlay[i];
420.                }
421.                StartCoroutine(CardsAnimated());
422.                Debug.Log("P1 Played " + msg);
423.                msg = "";
424.                played = true;
425.                Handler.lastPlay = Handler.turn;
426.            }
427.            IEnumerator CardsAnimated() {
428.                int temp = 0;
429.                foreach(Transform child in transform) {
430.                    if (temp == cardsToPlay.Count) {
431.                        break;
432.                    }
433.                    child.GetComponent < CardModel > ().TriggerDestroy();

434.                    temp++;
435.                }
436.                Instantiate(spinUp, gameObject.transform.position, Quatern
    ion.identity);
437.                yield
438.                return new WaitForSeconds(0.5 f);
439.                for (int x = 0; x < cardsToPlay.Count; x++) {
440.                    GameObject instance = Instantiate(cardPlayed, new Vect
    or3(0, 0.5 f * x - (cardsToPlay.Count * 0.25 f) + 0.25 f, -
    0.1 f * Handler.cardsOnField), Quaternion.Euler(0 f, 0 f, -90 f));
441.                    instance.GetComponent < CardModel > ().ToggleCard(card
    sToPlay[x]);
442.                    instance.tag = "CARDSONTHEFIELD";
443.                }
444.                cardsToPlay.Clear();
445.                yield
446.                return new WaitForSeconds(1.0 f);
447.                if (transform.childCount == 0) {
448.                    yield
```

```
449.                 return new WaitForSeconds(0.5 f);
450.                 win();
451.             }
452.             Handler.turn++;
453.             played = false;
454.         }
455.         public void win() {
456.             GameObject.Find("Handler").GetComponent < Handler > ().bgS
    ound.Stop();
457.             lose.Play();
458.             Debug.Log("P1 WON");
459.             GameObject winner = Instantiate(GameObject.Find("Handler")
    .GetComponent < Handler > ().WinnerPopUp, GameObject.Find("Handler").Ge
    tComponent < Handler > ().WinnerPopUp.transform.position, Quaternion.id
    entity);
460.             winner.transform.GetChild(3).GetComponent < Text > ().text
     = "P1";
461.             Time.timeScale = 0.0 f;
462.         }
463.     }
```

Kode Sumber A.1 *Computer Player* Greedy *Singleplayer*

[Halaman ini sengaja dikosongkan]

LAMPIRAN B

```
1.  using System.Collections;
2.  using System.Collections.Generic;
3.  using UnityEngine;
4.  using UnityEngine.UI;
5.  using UnityEngine.Networking;
6.  using System;
7.  public class AIMP: NetworkBehaviour {
8.      public bool populateHands = false;
9.      public bool handShown = false;
10.     public List < int > handCards = new List < int > ();
11.     public List < int > cardsToPlay = new List < int > ();
12.     public GameObject cardsMP;
13.     public GameObject WinNotif;
14.     public bool played = false;
15.     public AudioSource cardSpin; // Use this for initialization
16.     void Start() {
17.             if (transform.position == GameObject.Find("StartPosP3").tra
    nsform.position) {
18.                 transform.name = "P3";
19.             } else if (transform.position == GameObject.Find("StartPosP
    4").transform.position) {
20.                 transform.name = "P4";
21.             }
22.             if (transform.name == "P3") {
23.                 for (int x = 26; x < 39; x++) {
24.                     handCards.Add(GameObject.Find("ServerHandler").GetC
    omponent < ServerHandler > ().deckStack[x]);
25.                     if (GameObject.Find("ServerHandler").GetComponent <
     ServerHandler > ().deckStack[x] == 0) {
26.                         CmdChangeTurn(3);
27.                     }
28.                 }
29.                 handCards.Sort();
30.                 for (int y = 0; y < 13; y++) {
31.                     GameObject GOcardsMP = Instantiate(cardsMP, new Vec
    tor3(transform.position.x + 0.5 f * y -
     3, transform.position.y, transform.position.z -
     0.01 f * y), Quaternion.Euler(0 f, 0 f, 180 f), transform);
32.                 }
33.             } else if (transform.name == "P4") {
```

```
34.                 for (int x = 39; x < 52; x++) {
35.                     handCards.Add(GameObject.Find("ServerHandler").GetC
    omponent < ServerHandler > ().deckStack[x]);
36.                     if (GameObject.Find("ServerHandler").GetComponent <
     ServerHandler > ().deckStack[x] == 0) {
37.                         CmdChangeTurn(4);
38.                     }
39.                 }
40.                 handCards.Sort();
41.                 for (int y = 0; y < 13; y++) {
42.                     GameObject GOcardsMP = Instantiate(cardsMP, new Vec
    tor3(transform.position.x, transform.position.y + 0.5 f * y -
     3, transform.position.z -
     0.01 f * y), Quaternion.Euler(0 f, 0 f, 90 f), transform);
43.                 }
44.             }
45.         } // Update is called once per frame
46.     void Update() {
47.         if (!isServer || played || !GameObject.Find("ServerHandler").Ge
    tComponent < ServerHandler > ().gameStart) {
48.             return;
49.         }
50.         if (GameObject.Find("VariableHandler").GetComponent < VariableH
    andler > ().passedPlayer.Count == 3) {
51.             CmdChangeTurn(GameObject.Find("VariableHandler").GetCompone
    nt < VariableHandler > ().lastPlay);
52.             CmdAddPassedPlayer("RESET");
53.             CmdChangeCardPacket("Free");
54.         }
55.         if (GameObject.Find("VariableHandler").GetComponent < VariableH
    andler > ().passedPlayer.Contains(transform.name)) {
56.             passTurn();
57.         }
58.         int currentTurn = GameObject.Find("VariableHandler").GetCompone
    nt < VariableHandler > ().turn;
59.         string cardPacket = GameObject.Find("VariableHandler").GetCompo
    nent < VariableHandler > ().cardPacketMP;
60.         if ((transform.name == "P3" && currentTurn == 3) || (transform.
    name == "P4" && currentTurn == 4)) {
61.             if (cardPacket == "Free") {
62.                 cardsToPlay.Add(handCards[0]);
63.                 toField();
64.                 CmdChangeCardPacket("Single");
```

```
65.             } else if (cardPacket == "Single") {
66.                 bool play = false;
67.                 for (int x = 0; x < handCards.Count; x++) {
68.                     if (handCards[x] > GameObject.Find("VariableHandler
    ").GetComponent < VariableHandler > ().fieldCards[0]) {
69.                         play = true;
70.                         cardsToPlay.Add(handCards[x]);
71.                         toField();
72.                         break;
73.                     }
74.                 }
75.                 if (!play) {
76.                     passTurn();
77.                 }
78.             } else if (cardPacket == "Pair" && handCards.Count >= 2) {

79.                 if (findPair()) {
80.                     toField();
81.                 } else {
82.                     passTurn();
83.                 }
84.             } else if (cardPacket == "Threes" && handCards.Count >= 3)
    {
85.                 if (findThrees()) {
86.                     toField();
87.                 } else {
88.                     passTurn();
89.                 }
90.             } else if (cardPacket == "Straight" && handCards.Count >= 5
    ) {
91.                 if (findStraight()) {
92.                     toField();
93.                 } else if (findFlush()) {
94.                     toField();
95.                     CmdChangeCardPacket("Flush");
96.                 } else if (findFullHouse()) {
97.                     toField();
98.                     CmdChangeCardPacket("FullHouse");
99.                 } else if (findFourOfAKind()) {
100.                        toField();
101.                        CmdChangeCardPacket("FourOfAKind");
102.                    } else if (findStraightFlush()) {
103.                        toField();
```

```
104.                         CmdChangeCardPacket("StraightFlush");
105.                     } else if (findRoyalStraightFlush()) {
106.                         toField();
107.                         CmdChangeCardPacket("RoyalStraightFlush");
108.                     } else {
109.                         passTurn();
110.                     }
111.                 } else if (cardPacket == "Flush") {
112.                     if (findFlush()) {
113.                         toField();
114.                     } else if (findFullHouse()) {
115.                         toField();
116.                         CmdChangeCardPacket("FullHouse");
117.                     } else if (findFourOfAKind()) {
118.                         toField();
119.                         CmdChangeCardPacket("FourOfAKind");
120.                     } else if (findStraightFlush()) {
121.                         toField();
122.                         CmdChangeCardPacket("StraightFlush");
123.                     } else if (findRoyalStraightFlush()) {
124.                         toField();
125.                         CmdChangeCardPacket("RoyalStraightFlush");
126.                     } else {
127.                         passTurn();
128.                     }
129.                 } else if (cardPacket == "FullHouse") {
130.                     if (findFullHouse()) {
131.                         toField();
132.                     } else if (findFourOfAKind()) {
133.                         toField();
134.                         CmdChangeCardPacket("FourOfAKind");
135.                     } else if (findStraightFlush()) {
136.                         toField();
137.                         CmdChangeCardPacket("StraightFlush");
138.                     } else if (findRoyalStraightFlush()) {
139.                         toField();
140.                         CmdChangeCardPacket("RoyalStraightFlush");
141.                     } else {
142.                         passTurn();
143.                     }
144.                 } else if (cardPacket == "FourOfAKind") {
145.                     if (findFourOfAKind()) {
146.                         toField();
```

```
147.                     } else if (findStraightFlush()) {
148.                         toField();
149.                         CmdChangeCardPacket("StraightFlush");
150.                     } else if (findRoyalStraightFlush()) {
151.                         toField();
152.                         CmdChangeCardPacket("RoyalStraightFlush");
153.                     } else {
154.                         passTurn();
155.                     }
156.                 } else if (cardPacket == "StraightFlush") {
157.                     if (findStraightFlush()) {
158.                         toField();
159.                     } else if (findRoyalStraightFlush()) {
160.                         toField();
161.                         CmdChangeCardPacket("RoyalStraightFlush");
162.                     } else {
163.                         passTurn();
164.                     }
165.                 } else if (cardPacket == "RoyalStraightFlush") {
166.                     passTurn();
167.                 }
168.             }
169.         }[Command] public void CmdChangeCardPacket(string cardPacket)
    {
170.             RpcChangeCardPacket(cardPacket);
171.         }[ClientRpc] public void RpcChangeCardPacket(string cardPacket
    ) {
172.             GameObject.Find("VariableHandler").GetComponent < Variable
    Handler > ().cardPacketMP = cardPacket;
173.         }[Command] void CmdChangeTurn(int nextTurn) {
174.             RpcChangeTurn(nextTurn);
175.         }[ClientRpc] void RpcChangeTurn(int nextTurn) {
176.             GameObject.Find("VariableHandler").GetComponent < Variable
    Handler > ().turn = nextTurn;
177.         }[Command] public void CmdSpinCards(GameObject GOcardsMP) {
178.             RpcSpinCards(GOcardsMP);
179.         }[ClientRpc] public void RpcSpinCards(GameObject GOcardsMP) {

180.             Instantiate(GOcardsMP);
181.         }[Command] void CmdClearFieldCards() {
182.             RpcClearFieldCards();
183.         }[ClientRpc] void RpcClearFieldCards() {
```

```
184.             GameObject.Find("VariableHandler").GetComponent < Variable
    Handler > ().fieldCards.Clear();
185.         }[Command] void CmdUpdateFieldCards(int cardNumber) {
186.             RpcUpdateFieldCards(cardNumber);
187.         }[ClientRpc] void RpcUpdateFieldCards(int cardNumber) {
188.             GameObject.Find("VariableHandler").GetComponent < Variable
    Handler > ().fieldCards.Add(cardNumber);
189.         }
190.         public void toField() {
191.             if (played) {
192.                 return;
193.             }
194.             played = true;
195.             StartCoroutine(toFieldAnimations());
196.         }[Command] public void CmdDestroyCards(string playerName, int
    cardIndex) {
197.             RpcDestroyCards(playerName, cardIndex);
198.         }[ClientRpc] public void RpcDestroyCards(string playerName, in
    t cardIndex) {
199.             GameObject.Find(playerName).transform.GetChild(cardIndex).
    GetComponent < CardModelMP > ().TriggerDestroy();
200.         }[Command] public void CmdAnimationCards() {
201.             RpcAnimationCards();
202.         }[ClientRpc] public void RpcAnimationCards() {
203.             if (GameObject.Find("VariableHandler").GetComponent < Vari
    ableHandler > ().turn == 3) {
204.                 cardSpin.Play();
205.                 Instantiate(GameObject.Find("ServerHandler").GetCompon
    ent < ServerHandler > ().spinDown, GameObject.Find("P3").transform.posi
    tion, Quaternion.identity);
206.             } else if (GameObject.Find("VariableHandler").GetComponent
     < VariableHandler > ().turn == 4) {
207.                 cardSpin.Play();
208.                 Instantiate(GameObject.Find("ServerHandler").GetCompon
    ent < ServerHandler > ().spinLeft, GameObject.Find("P4").transform.posi
    tion, Quaternion.identity);
209.             }
210.         }[Command] public void CmdPutCardsOnField(Vector3 tempPos, int
     cardIndex, Quaternion tempRot) {
211.             RpcPutCardsOnField(tempPos, cardIndex, tempRot);
212.         }[ClientRpc] public void RpcPutCardsOnField(Vector3 tempPos, i
    nt cardIndex, Quaternion tempRot) {
213.             handCards.Remove(cardIndex);
```

```
214.                GameObject instance = Instantiate(cardsMP, tempPos, tempRo
    t);
215.                instance.GetComponent < CardModelMP > ().ToggleCard(cardIn
    dex);
216.                instance.tag = "CARDSONTHEFIELD";
217.            }[Command] public void CmdUpdateLastPlay(int playerIndex) {
218.                RpcUpdateLastPlay(playerIndex);
219.            }[ClientRpc] public void RpcUpdateLastPlay(int playerIndex) {

220.                GameObject.Find("VariableHandler").GetComponent < Variable
    Handler > ().lastPlay = playerIndex;
221.            }[Command] public void CmdCOFCount(int newCount) {
222.                RpcCOFCount(newCount);
223.            }[ClientRpc] public void RpcCOFCount(int newCount) {
224.                GameObject.Find("VariableHandler").GetComponent < Variable
    Handler > ().COFCount = newCount;
225.            }[Command] public void CmdWin(string playerName) {
226.                RpcWin(playerName);
227.            }[ClientRpc] public void RpcWin(string playerName) {
228.                GameObject winner = Instantiate(WinNotif, WinNotif.transfo
    rm.position, Quaternion.identity);
229.                winner.transform.GetChild(3).GetComponent < Text > ().text
     = playerName;
230.                winner.transform.GetChild(4).GetComponent < Text > ().text
     = "You Lose";
231.                Time.timeScale = 0.0 f;
232.            }
233.            IEnumerator toFieldAnimations() {
234.                CmdClearFieldCards();
235.                cardsToPlay.Sort();
236.                for (int x = 0; x < cardsToPlay.Count; x++) {
237.                    CmdUpdateFieldCards(cardsToPlay[x]);
238.                    CmdDestroyCards(transform.name, 0);
239.                }
240.                CmdAnimationCards();
241.                yield
242.                return new WaitForSeconds(0.5 f);
243.                int COFC = GameObject.Find("VariableHandler").GetComponent
    < VariableHandler > ().COFCount;
244.                if (transform.name == "P3") {
245.                    for (int x = 0; x < cardsToPlay.Count; x++) {
246.                        COFC++;
```

```
247.                     Vector3 tempPos = new Vector3(0.5 f * x -
    (cardsToPlay.Count * 0.25 f) + 0.25 f, 0, -0.1 f * COFC);
248.                     Quaternion temRot = Quaternion.Euler(0 f, 0 f, 180
    f);
249.                     CmdPutCardsOnField(tempPos, cardsToPlay[x], temRot
    );
250.                 }
251.                 CmdUpdateLastPlay(3);
252.             } else if (transform.name == "P4") {
253.                 for (int x = 0; x < cardsToPlay.Count; x++) {
254.                     COFC++;
255.                     Vector3 tempPos = new Vector3(0, 0.5 f * x -
    (cardsToPlay.Count * 0.25 f) + 0.25 f, -0.1 f * COFC);
256.                     Quaternion temRot = Quaternion.Euler(0 f, 0 f, -
    90 f);
257.                     CmdPutCardsOnField(tempPos, cardsToPlay[x], temRot
    );
258.                 }
259.                 CmdUpdateLastPlay(4);
260.             }
261.             CmdCOFCount(COFC);
262.             cardsToPlay.Clear();
263.             yield
264.             return new WaitForSeconds(0.5 f);
265.             if (handCards.Count == 0) {
266.                 CmdWin(transform.name);
267.             }
268.             yield
269.             return new WaitForSeconds(0.5 f);
270.             if (GameObject.Find("VariableHandler").GetComponent < Vari
    ableHandler > ().turn < 4) {
271.                 CmdChangeTurn(GameObject.Find("VariableHandler").GetCo
    mponent < VariableHandler > ().turn + 1);
272.             } else {
273.                 CmdChangeTurn(1);
274.             }
275.             played = false;
276.         }[Command] public void CmdAddPassedPlayer(string playerName) {

277.             RpcAddPassedPlayer(playerName);
278.         }[ClientRpc] public void RpcAddPassedPlayer(string playerName)
    {
279.             if (playerName == "RESET") {
```

```
280.                 GameObject.Find("VariableHandler").GetComponent < Vari
    ableHandler > ().passedPlayer.Clear();
281.                 GameObject[] DestroyCOF = GameObject.FindGameObjectsWi
    thTag("CARDSONTHEFIELD");
282.                 foreach(GameObject destroy in DestroyCOF) {
283.                     Destroy(destroy);
284.                 }
285.             } else {
286.                 GameObject.Find("VariableHandler").GetComponent < Vari
    ableHandler > ().passedPlayer.Add(playerName);
287.             }
288.         }
289.         public void passTurn() {
290.             if (GameObject.Find("VariableHandler").GetComponent < Vari
    ableHandler > ().cardPacketMP != "Free") {
291.                 if (!GameObject.Find("VariableHandler").GetComponent <
     VariableHandler > ().passedPlayer.Contains(transform.name)) {
292.                     CmdAddPassedPlayer(transform.name);
293.                 } else {
294.                     if (GameObject.Find("VariableHandler").GetComponen
    t < VariableHandler > ().turn < 4) {
295.                         CmdChangeTurn(GameObject.Find("VariableHandler
    ").GetComponent < VariableHandler > ().turn + 1);
296.                     } else {
297.                         CmdChangeTurn(1);
298.                     }
299.                 }
300.             }
301.         }
302.         public bool findPair() {
303.             for (int x = 1; x < handCards.Count; x++) {
304.                 if (handCards[x -
     1] / 4 == handCards[x] / 4 && handCards[x] > Mathf.Max(GameObject.Find
    ("VariableHandler").GetComponent < VariableHandler > ().fieldCards.ToAr
    ray())) {
305.                     cardsToPlay.Add(handCards[x - 1]);
306.                     cardsToPlay.Add(handCards[x]);
307.                     return true;
308.                 }
309.             }
310.             return false;
311.         }
312.         public bool findThrees() {
```

```
313.             for (int x = 2; x < handCards.Count; x++) {
314.                 if (handCards[x -
    2] / 4 == handCards[x] / 4 && handCards[x -
    1] / 4 == handCards[x] / 4 && handCards[x] > Mathf.Max(GameObject.Find
   ("VariableHandler").GetComponent < VariableHandler > ().fieldCards.ToAr
   ray())) {
315.                     cardsToPlay.Add(handCards[x - 2]);
316.                     cardsToPlay.Add(handCards[x - 1]);
317.                     cardsToPlay.Add(handCards[x]);
318.                     return true;
319.                 }
320.             }
321.             return false;
322.         }
323.         public bool findStraight() {
324.             string cardPacket = GameObject.Find("VariableHandler").Get
   Component < VariableHandler > ().cardPacketMP;
325.             if (cardPacket != "Straight") {
326.                 return false;
327.             }
328.             for (int x = 0; x < handCards.Count - 4; x++) {
329.                 cardsToPlay.Add(handCards[x]);
330.                 for (int y = x + 1; y < handCards.Count; y++) {
331.                     if (handCards[y] / 4 == Mathf.Max(cardsToPlay.ToAr
   ray()) / 4 + 1) {
332.                         cardsToPlay.Add(handCards[y]);
333.                     }
334.                     if (cardsToPlay.Count == 5 || handCards[y] / 4 > M
   athf.Max(cardsToPlay.ToArray()) / 4 + 1) {
335.                         break;
336.                     }
337.                 }
338.                 if (cardsToPlay.Count == 5 && Mathf.Max(cardsToPlay.To
   Array()) > Mathf.Max(GameObject.Find("VariableHandler").GetComponent <
   VariableHandler > ().fieldCards.ToArray())) {
339.                     bool temp = false;
340.                     for (int z = 0; z < cardsToPlay.Count -
    1; z++) {
341.                         if (cardsToPlay[z] % 4 != cardsToPlay[z + 1] %
    4) {
342.                             temp = true;
343.                             break;
344.                         }
```

```
345.                     }
346.                     if (temp) {
347.                         return true;
348.                     } else {
349.                         cardsToPlay.Clear();
350.                     }
351.                 } else {
352.                     cardsToPlay.Clear();
353.                 }
354.             }
355.             return false;
356.         }
357.         public bool findFlush() {
358.             string cardPacket = GameObject.Find("VariableHandler").Get
    Component < VariableHandler > ().cardPacketMP;
359.             for (int x = 0; x < handCards.Count - 4; x++) {
360.                 cardsToPlay.Add(handCards[x]);
361.                 for (int y = x + 1; y < handCards.Count; y++) {
362.                     if (handCards[y] % 4 == cardsToPlay[0] % 4) {
363.                         cardsToPlay.Add(handCards[y]);
364.                     }
365.                     if (cardsToPlay.Count == 5) {
366.                         break;
367.                     }
368.                 }
369.                 if (cardsToPlay.Count == 5) {
370.                     if (cardPacket != "Flush" || (cardPacket == "Flush
    " && Mathf.Max(cardsToPlay.ToArray()) > Mathf.Max(GameObject.Find("Vari
    ableHandler").GetComponent < VariableHandler > ().fieldCards.ToArray())
    )) {
371.                         bool temp = false;
372.                         for (int z = 0; z < cardsToPlay.Count -
     1; z++) {
373.                             if ((cardsToPlay[z] / 4) + 1 != cardsToPla
    y[z + 1] / 4) {
374.                                 temp = true;
375.                                 break;
376.                             }
377.                         }
378.                         if (temp) {
379.                             return true;
380.                         } else {
381.                             cardsToPlay.Clear();
```

```
382.                         }
383.                     } else {
384.                         cardsToPlay.Clear();
385.                     }
386.                 } else {
387.                     cardsToPlay.Clear();
388.                 }
389.             }
390.             return false;
391.         }
392.         public bool findFullHouse() {
393.             string cardPacket = GameObject.Find("VariableHandler").Get
    Component < VariableHandler > ().cardPacketMP;
394.             for (int x = 0; x < handCards.Count - 2; x++) {
395.                 if (handCards[x] / 4 == handCards[x + 1] / 4 && handCa
    rds[x] / 4 == handCards[x + 2] / 4) {
396.                     cardsToPlay.Add(handCards[x]);
397.                     cardsToPlay.Add(handCards[x + 1]);
398.                     cardsToPlay.Add(handCards[x + 2]);
399.                     if (cardPacket == "FullHouse") {
400.                         if (GameObject.Find("VariableHandler").GetComp
    onent < VariableHandler > ().fieldCards[1] / 4 == GameObject.Find("Vari
    ableHandler").GetComponent < VariableHandler > ().fieldCards[2] / 4) {

401.                             if (cardsToPlay[0] > GameObject.Find("Vari
    ableHandler").GetComponent < VariableHandler > ().fieldCards[0]) {
402.                                 break;
403.                             } else {
404.                                 cardsToPlay.Clear();
405.                             }
406.                         } else {
407.                             if (cardsToPlay[0] > GameObject.Find("Vari
    ableHandler").GetComponent < VariableHandler > ().fieldCards[4]) {
408.                                 break;
409.                             } else {
410.                                 cardsToPlay.Clear();
411.                             }
412.                         }
413.                     } else {
414.                         break;
415.                     }
416.                 }
417.             }
```

```
418.             if (cardsToPlay.Count < 3) {
419.                 return false;
420.             }
421.             for (int x = 0; x < handCards.Count - 1; x++) {
422.                 if (handCards[x] / 4 == handCards[x + 1] / 4 && handCa
    rds[x] / 4 != cardsToPlay[0] / 4) {
423.                     cardsToPlay.Add(handCards[x]);
424.                     cardsToPlay.Add(handCards[x + 1]);
425.                     break;
426.                 }
427.             }
428.             if (cardsToPlay.Count < 5) {
429.                 return false;
430.             }
431.             return true;
432.         }
433.         public bool findFourOfAKind() {
434.             string cardPacket = GameObject.Find("VariableHandler").Get
    Component < VariableHandler > ().cardPacketMP;
435.             for (int x = 0; x < handCards.Count - 3; x++) {
436.                 if (handCards[x] / 4 == handCards[x + 1] / 4 && handCa
    rds[x] / 4 == handCards[x + 2] / 4 && handCards[x] / 4 == handCards[x +
     3] / 4) {
437.                     cardsToPlay.Add(handCards[x]);
438.                     cardsToPlay.Add(handCards[x + 1]);
439.                     cardsToPlay.Add(handCards[x + 2]);
440.                     cardsToPlay.Add(handCards[x + 3]);
441.                     if (cardPacket == "FourOfAKind") {
442.                         if (GameObject.Find("VariableHandler").GetComp
    onent < VariableHandler > ().fieldCards[0] / 4 == GameObject.Find("Vari
    ableHandler").GetComponent < VariableHandler > ().fieldCards[1] / 4) {

443.                             if (cardsToPlay[0] > GameObject.Find("Vari
    ableHandler").GetComponent < VariableHandler > ().fieldCards[0]) {
444.                                 break;
445.                             } else {
446.                                 cardsToPlay.Clear();
447.                             }
448.                         } else {
449.                             if (cardsToPlay[0] > GameObject.Find("Vari
    ableHandler").GetComponent < VariableHandler > ().fieldCards[4]) {
450.                                 break;
451.                             } else {
```

```
452.                                 cardsToPlay.Clear();
453.                             }
454.                         }
455.                     } else {
456.                         break;
457.                     }
458.                 }
459.             }
460.             if (cardsToPlay.Count < 4) {
461.                 return false;
462.             }
463.             for (int x = 0; x < handCards.Count; x++) {
464.                 if (handCards[x] / 4 != cardsToPlay[0] / 4) {
465.                     cardsToPlay.Add(handCards[x]);
466.                     break;
467.                 }
468.             }
469.             if (cardsToPlay.Count < 5) {
470.                 return false;
471.             }
472.             return true;
473.         }
474.         public bool findStraightFlush() {
475.             for (int x = 0; x < handCards.Count - 4; x++) {
476.                 cardsToPlay.Add(handCards[x]);
477.                 for (int y = x + 1; y < handCards.Count; y++) {
478.                     if (handCards[y] / 4 == Mathf.Max(cardsToPlay.ToAr
    ray()) / 4 + 1) {
479.                         cardsToPlay.Add(handCards[y]);
480.                     }
481.                     if (cardsToPlay.Count == 5 || handCards[y] / 4 > M
    athf.Max(cardsToPlay.ToArray()) / 4 + 1) {
482.                         break;
483.                     }
484.                 }
485.                 if (cardsToPlay.Count == 5) {
486.                     bool temp = true;
487.                     for (int z = 0; z < cardsToPlay.Count -
     1; z++) {
488.                         if (cardsToPlay[z] % 4 != cardsToPlay[z + 1] %
     4) {
489.                             temp = false;
490.                             break;
```

```
491.                         }
492.                     }
493.                     if (temp) {
494.                         if (cardsToPlay[0] % 4 > GameObject.Find("Vari
    ableHandler").GetComponent < VariableHandler > ().fieldCards[0] % 4) {

495.                             return true;
496.                         } else if (cardsToPlay[0] % 4 == GameObject.Fi
    nd("VariableHandler").GetComponent < VariableHandler > ().fieldCards[0]
     % 4) {
497.                             if (Mathf.Max(cardsToPlay.ToArray()) > Mat
    hf.Max(GameObject.Find("VariableHandler").GetComponent < VariableHandle
    r > ().fieldCards.ToArray())) {
498.                                 return true;
499.                             } else {
500.                                 cardsToPlay.Clear();
501.                             }
502.                         } else {
503.                             cardsToPlay.Clear();
504.                         }
505.                     } else {
506.                         cardsToPlay.Clear();
507.                     }
508.                 } else {
509.                     cardsToPlay.Clear();
510.                 }
511.             }
512.             return false;
513.         }
514.         public bool findRoyalStraightFlush() {
515.             if (handCards.Contains(47) && handCards.Contains(43) && ha
    ndCards.Contains(39) && handCards.Contains(35) && handCards.Contains(31
    )) {
516.                 cardsToPlay.Add(47);
517.                 cardsToPlay.Add(43);
518.                 cardsToPlay.Add(39);
519.                 cardsToPlay.Add(35);
520.                 cardsToPlay.Add(31);
521.                 return true;
522.             } else {
523.                 return false;
524.             }
525.         }
```

```
526.     }
```

Kode Sumber B.2 *Computer Player* Greedy *Multiplayer*